

# CasFinder System Documentation: Strategy, Installation, Usage, and Performance

Supporting Information for  
“CasFinder: Flexible algorithm for identifying specific Cas9 targets in genomes”

John Aach, Prashant Mali, George M. Church\*

Department of Genetics, Harvard Medical School, Boston, MA 02115, USA

\* Corresponding author

## Contents

<a href="#">CasFinder design</a> .....	3
<a href="#">CasFinder design overview</a> .....	3
<a href="#">Technical note: Why queries combining targeting sequence and PAM are difficult</a> .....	4
<a href="#">Overview of the CasFinder system of programs</a> .....	6
<a href="#">CasFinder system availability and citation</a> .....	7
<a href="#">CasFinder system installation and customization</a> .....	8
<a href="#">1. Installing CasFinder program components</a> .....	8
<a href="#">2. Preparing genome files needed for CasFinder operation</a> .....	8
<a href="#">3. Editing the CasFinder system configuration files</a> .....	9
<a href="#">4. Testing the installation</a> .....	13
<a href="#">General features of CasFinder and CasValue operation</a> .....	14
<a href="#">Program option attributes and coordination</a> .....	14
<a href="#">Input sequence processing options</a> .....	17
<a href="#">CasFinder operation</a> .....	18
<a href="#">CasFinder program options</a> .....	18
<a href="#">CasFinder input file specifications</a> .....	20
<a href="#">CasFinder output files</a> .....	22
<a href="#">CasValue operation</a> .....	26
<a href="#">CasValue program options</a> .....	26
<a href="#">CasValue input file specifications</a> .....	28
<a href="#">CasValue output files</a> .....	30
<a href="#">Sensitivity, specificity, and FDR of specific Cas9 site detection: Preliminary assessment</a> .....	33
<a href="#">Using CasValue to re-screen CasFinder targets for higher specificity</a> .....	36
<a href="#">CasFinder system performance considerations</a> .....	40
<a href="#">Generation of the all human and mouse exome catalogs of Cas9 sites</a> .....	42
<a href="#">Definition of the human and mouse exomes</a> .....	43
<a href="#">Generation of the Cas9 site catalogs for each exome</a> .....	44
<a href="#">Generation of reduced set of Cas9 targets for gene knockout screens</a> .....	44
<a href="#">Statistics at the gene copy level</a> .....	46
<a href="#">Comparison of the CasFinder system with other Cas9 target evaluation algorithms</a> .....	46
<a href="#">References</a> .....	49

## Tables and Figures

<a href="#">Table S1: Components of the CasFinder system of programs .....</a>	7
<a href="#">Table S2: CasFinder program options .....</a>	19
<a href="#">Table S3: CasValue program options .....</a>	28
<a href="#">Table S4: Cas9-targeted sites characterized for off-targets in human genome .....</a>	34
<a href="#">Table S5: Use of CasValue to screen sites for higher specificity .....</a>	38
<a href="#">Table S6: Numbers of sites accepted at default settings rejected at smaller seed lengths .....</a>	40
<a href="#">Table S7: Numbers of gene copies with targets that do not appear in reduced target sets .....</a>	46
<a href="#">Table S8: Comparison of CasFinder system with other Cas9 target evaluation algorithms .....</a>	47
<a href="#">Figure S3: Excerpt of CASFINDER_CONFIG.txt file at authors' installation .....</a>	10
<a href="#">Figure S4: Excerpt of distributed CASFINDER_CAS9_CONFIG.txt .....</a>	13
<a href="#">Figure S5: Fraction of inaccurately evaluated SP Cas9 sites at different bowtie -k values .....</a>	24

(Note: Figures S1 and S2 are provided in separate Supporting Information documents.)

### *CasFinder design overview*

As described in the main article, in type II prokaryotic Clustered Regularly Interspaced Palindromic Repeats (CRISPR) systems, a complex of a Cas9 protein with a pair of RNAs recognizes a DNA target that consists of a short stretch of DNA complementary to the 5' end of one of the RNAs followed by a Cas9-specific Protospacer Adjacent Motif (PAM). In applications, the two RNAs are frequently combined into a single guide RNA (sgRNA) whose 5'-most 20bp comprises the targeting region. The initial characterization of Cas9 specificity suggested that mismatches between the last 12-13nt of the RNA targeting region (the "seed" region) and DNA targets would abrogate Cas9 activity [4]. This led to the following simple and efficient algorithm for identifying putatively specific Cas9 targets in a genome: For any given candidate 20bp sequence followed by a PAM motif, all one need do is determine whether there is an exactly matching seed sequence followed by a PAM in the genome elsewhere than the candidate itself, and reject any candidate for which one is found as 'non-specific'. In our initial work in ref. [3], we rapidly and easily implemented this algorithm for *S. pyogenes* (SP) Cas9, by extracting all 20nt candidate Cas9 target sites 5'-NNNNN NNBBB BBBB BBBB-NGG-3' within the sequences of interest (where the B(13) region was taken as the seed sequence within the site and the final NGG is the SP Cas9 PAM), and using bowtie [5] to find exact matches in the genome to the four query sequences B(13)AGG, B(13)CGG, B(13)GGG, and B(13)TGG. The need for four queries arose because bowtie does not support 'wildcard' N bases in queries, but performance was still very good because bowtie is extremely fast. Note that while the length of the seed region is not settled, for practical reasons described in the article text we used 13bp seeds for human genome queries (and continue to do so here). However, the present understanding of specificity requires that seed mismatches, non-seed matches, and multiple and sometimes complex PAMs be taken into account (see main article for references), and this greatly complicates this strategy (for details, see the [Technical note](#) below). Thus, in our present method, we forewent the performance advantages of combined seed and PAM queries, in favor of a simple, practical, and highly flexible system with the following design:

1. The system implements two distinct functions: (i) finding candidate Cas9 sites within user-specified sequences, performed by the CasFinder program, and (ii) evaluating candidate target sequences in a genome, performed by CasValue. CasFinder calls CasValue to evaluate the candidates it finds, but CasValue may be called independently and can be used to evaluate the specificity of Cas9 targets found by other systems.
2. Because an increasing number of Cas9s are coming into use and uncertainty still exists regarding the structural features of their Cas9 sites, Cas9 target definitions are maintained entirely in user-editable configuration files separate from program code. Cas9s defined to the system have the following properties: (i) a target sequence length, (ii) a start and end point of a seed region within the target sequence, (iii) a list of possible primary PAM sequences that are specifiable using standard degenerate base codes, (iv) a similar list of possible secondary PAM sequences.
3. CasValue evaluates lists of input target sequences of the form  $N(L) = 5'-N(u)B(s)N(d)-3'$  for the presence of potential off-targets in a genome, where  $L$  is the target length and  $s$  is the seed length defined for the Cas9 it is processing, and  $N(u)$  and  $N(d)$  are upstream and downstream non-seed sequences. CasValue starts by (i) using bowtie to find all matches to  $B(s)$  seed sequences in a specified genome with up to a specific number of mismatches, and then it (ii) retrieves from the genome sequence the full  $5'-N'(u)B'(s)N'(d)-P(k)-3'$  Cas9 footprints around each match, where  $k$  is the Cas9's maximum PAM length. (iii) Matching of the  $P(k)$  against

primary or secondary PAMs is checked *via* regular expression pattern matches, and sites for which no match is found are discarded. (iv) For remaining footprints, a score is computed as a weighted sum of the numbers of mismatches between the two N' and N regions, and of the B' and B regions (where different weights may be applied to the N and B regions), and an additional cost is added if the P(k) matches a secondary but not a primary PAM. (v) CasValue is further provided an exclusion parameter  $-x$  that defines the upper limit of scores of matches it will consider as potential off-targets, and ignores scores  $> -x$ . (v) Finally, each input target sequence has an 'accept' number that indicates the number of genome matches with score  $\leq -x$  that the input sequence may be allowed to have without being rejected as a non-specific candidate target. CasValue reports all input candidates that have not been rejected. All scoring parameters are under user control but default to 1 for each mismatch (for both seed and non-seed), 1 for the alternate PAM cost, and 3 for  $-x$ . The 'accept' value defaults to 1, the match limit appropriate to specific genome-endogenous targets (where the single accepted match would be the endogenous target itself), but may be set to 0 to test exogenous sequence targets, or  $> 1$  for targets that may occur multiple times in homologs or otherwise similar sequences.

4. CasFinder is passed an input sequence file that may contain either full FASTA sequences, or FASTA headers alone (i.e., without appended sequence) that specify sequence location ranges and the keyword \*EXTRACT\*: For these, CasFinder extracts the location ranges directly from the genome sequence files also used in step 3 to extract Cas9 target footprints. In either case, CasFinder searches the specified sequences for occurrences of primary PAMs to identify candidate Cas9 targets, optionally eliminating those that contain long strings of Ts or repeatmasked sequence, and passes the candidate target sequences to CasValue for evaluation. CasFinder reports all candidates that CasValue does not reject as non-specific. Special keywords included in the FASTA headers of the input sequences allow target locations to be reported as offsets from a user-specified reference point (such as the 5' end of an exon or gene), to specify the sequence's 'accept' number, or whether candidates in repeatmasked sequence should be retained.

*Technical note: Why queries combining targeting sequence and PAM are difficult*

In this note, we explain why the complex picture of Cas9 specificity that has emerged from recent analyses [6-9], and the characterization of Cas9s from a number of different organisms [10], greatly complicates the use of queries that incorporate both targeting sequence and PAMs. The basic problem is that such queries only work efficiently when it is assumed that PAM sequences contain little degeneracy or ambiguity and where seed mismatches are not tolerated. Complications and scaling problems quickly arise when these assumptions are violated.

To see this, consider our initial specificity check developed in in ref. [3] (described above), whereby we searched for off-targets to candidate SP Cas9 target sites by using four bowtie queries B(13)AGG, B(13)CGG, B(13)GGG, and B(13)TGG to accommodate the single degenerate N position in the SP Cas9 NGG PAM. This method does not scale well when extended to more recently characterized Cas9s with more highly degenerate PAMs, such as that of the Cas9 of *N. meningitis* (NM), which is active at five PAMs all beginning NNNNG (see Table 1 of main article and [11]): Instead of requiring four queries to cover the degenerate region of the PAM, NM Cas9 would require 256 to evaluate each candidate Cas9 target. As an alternative, the need for multiple queries can be circumvented by using bowtie 2 [12] in place of bowtie, because bowtie 2 can be made to process wildcard Ns in queries using suitable parameter settings. However, this entails using bowtie 2 to process end-to-end alignments that are

scored *via* dynamic programming, and this incurs a substantial performance penalty: Indeed, we found that using bowtie 2 to process exact matches to a single B(13)NGG query with wildcard 'N' was ~25x slower in human genome searches than the method above of using bowtie to process the four queries B(13)AGG, B(13)CGG, B(13)GGG, and B(13)TGG (data not shown). Thus, either way, applying the initial strategy of ref. [3] to NM Cas9 leads to performance difficulties.

Complications also arise when using bowtie or bowtie 2 to return genome locations matching a combined seed and PAM where mismatches are allowed at the genome location, because the mismatching positions of the returned sequences can occur in fixed positions of the PAM as well as the seed. Focusing for concreteness on B(13)NGG queries needed for SP Cas9, consider the case of using bowtie to return locations with up to a single mismatch, and consider for now only one of the four needed queries: B(13)CGG. Genome sequences returned for this query could be of the form: (i) B'(13)CGG sequences where B'(13) differs from B(13) in a single position, (ii) B(13)DGG sequences (D = A, G, or T), (iii) B(13)CGH or B(13)CYG sequences (H = A, C, or T, and Y = C or T), or (iv) B(13)CAG sequences. Of these, form (iii) sequences are irrelevant because they do not contain PAMs recognized by the Cas9, eliminating these locations as candidate site off-targets, while the others indicate potential off-targets (where type (iv) locations have the secondary NAG PAM). But the sequences returned by B(13)CGG query also fail to exhaust the potential off-targets in the genome because B'(13)DGG sequences in the genome are not returned. To find them requires the other three bowtie queries B(13)AGG, B(13)GGG, and B(13)TGG, but the sequences returned for these will not only yield additional superfluous type (iii) sequences, but will also redundantly return type (ii) sequences that were seen by previous queries. Again, return of these extra superfluous and redundant type (ii) and (iii) sequences can be avoided by using bowtie 2 with a single B(13)NGG query, but only by incurring a substantial performance penalty. Finally, note that using bowtie to return matches with *more than one* mismatch fails to resolve these problems. For instance, although having bowtie return matches with two mismatches to the query B(13)CGG will now return the B'(13)DGG matches that were not seen when only matches with one mismatch were returned, this will result in the return of additional forms of irrelevant sequences such as B(13)DGH and B(13)CYH.

The fact that performance penalties of one or another sort will accrue by attempting to extend the strategy of ref. [3] of using queries that combine targeting sequence and PAM to exhaustively identify potential Cas9 targets, does not imply that the performance cannot be optimized for any particular Cas9. It is clear that the degree and kind of the performance impacts above depend largely on the structure of the PAM. For instance, for Cas9s with few wildcard positions, such as SP Cas9, four queries per candidate with bowtie are faster than a single wildcard query with bowtie 2, while for Cas9s with many wildcards like NM Cas9, the ~25-fold higher costs of bowtie 2 are faster than the bowtie approach, which would here require 256 queries per candidate. On the other hand, the number of non-wildcard positions in the PAM will determine the extent of superfluous retrievals of type (iii) resulting from multiple bowtie queries. For any particular Cas9 and genome, the trade-offs between these two approaches could in principal be characterized and an optimal strategy chosen. But the broader issue is that, even among the few Cas9s currently developed for use (see Table 1 in the main article and [11]), it is clear that PAMs span a wide range of size and degeneracy, so that it is unlikely that whatever strategy is optimal for queries based on combined targeting sequences and PAMs for one Cas9 will be optimal for another. Thus, in developing the CasFinder system, we made a strategic decision to simply abandon the use of combined queries.

Finally, we note that while the above difficulties are discussed with respect to bowtie and bowtie 2, they are very general and will apply equally well to other sequence mapping tools with similar features.

Fundamentally, the issue is that the kind of site scoring needed to evaluate Cas9 off-targeting within a genome is simply not a functionality that either is, or can be expected to be, supported by high-throughput sequence mapping tools alone. Ultimately, Cas9 off-target analysis requires that sites be evaluated differently for mismatching positions in the seed region, non-seed targeting sequence region, and for degenerate and fixed positions of the PAMs. In principle, position-specific weight matrices (PSSMs) could be used to compute scores for off-target potential: One possible PSSM scoring regime would feature four weights:  $W_s$  (seed regions),  $W_{ns}$  (non-seed targeting regions),  $W_N$  and  $W_A$  (non-allowed and allowed bases in PAM positions), where  $W_N \gg W_s > W_{ns} > W_A = 0$ , where low scores would indicate high off-target potential. However, no algorithm currently exists for rapid and comprehensive retrieval of all sites with low scoring PSSMs in a genome for an arbitrary query. In the absence of such an algorithm, sequence mapping tools provide a very rapid way to retrieve targeting sequence matches whose Cas9 footprints can be extracted and evaluated for the presence of a PAM. In the CasFinder system, we continue to use bowtie for the former purpose and now perform the footprint extraction and PAM evaluation in separate code. A possible direction for improving Cas9 site analysis generally would be to extend sequence mapping tools to incorporate footprint extraction and scoring directly.

### Overview of the CasFinder system of programs

The CasFinder system comprises a set of two perl programs and associated configuration files and perl packages whose purpose is to enable users to find and evaluate CRISPR/Cas9 target sites in user-provided sequence that are unlikely to have off-target sites in a user-specified genome. The components of the system are listed in [Table S1](#).

Program component	Current component file	Description
CasFinder	CasFinder.pl	Perl program that searches user-provided sequences for candidate Cas9 targets, passes them to CasValue for specificity analysis, and returns the Cas9 targets to deemed to be specific the user.
CasValue	CasValue_v2.pl	Perl program that reads a FASTA file of candidate Cas9 target sequences and identifies the subset meeting user-defined specificity constraints in a user-specified genome.
CASFINDER_CONFIG	CASFINDER_CONFIG.txt	User-editable text file in which installation-specific genome sequence and program resources needed by CasFinder and CasValue are defined.
CASFINDER_CAS9_CONFIG	CASFINDER_CAS9_CONFIG.txt	User-editable text file in which characteristics of target and PAM sequences of CRISPR/Cas9 systems are defined.
CasFinderConfigFile	CasFinderConfigFile.pm	Perl package used by CasFinder and CasValue as an interface to the CASFINDER_CONFIG file.
CasFinderConfigCas9File	CasFinderConfigCas9File.pm	Perl package used by CasFinder and CasValue as an interface to the

	CASFINDER_CAS9_CONFIG file.
--	-----------------------------

**Table S1:** Components of the CasFinder system of programs

The CasFinder system is designed to be flexible. After system installation, users can add or change the definitions of Cas9 targets processable by the system simply by editing a text configuration file without having to modify program code. Users can similarly add a genome to the system simply by obtaining or generating a set of organism-specific sequence files and declaring them in a second text configuration file, again without having to modify program code. Details on how Cas9 sites are evaluated for specificity can be changed and controlled by combinations of configuration file, program, and input file parameters, so that, again, program modifications are unnecessary. Finally, the system can either find Cas9 targets in user-specified sequence, which sites are then evaluated for specificity, or specificity can be evaluated for targets generated by other software systems.

The CasFinder system also provides several features designed to enhance usability. For genomes defined to the system, users can simply specify the sequence ranges they wish to have the system analyze without including the sequence itself. The user can also provide complete sequence input, including exogenous sequence, and have the system find targets in the exogenous sequence that do not have off-targets in the user-specified genome. Users can define a reference location within each (provided or specified) input sequence and have the system compute offsets of each target relative to the reference location, making it easy to (e.g.) identify the targets that are closest to the 5' end of a gene or exon. Finally, CasValue's specificity evaluation computes a score for each site that can be used to pick targets estimated to be more specific than others (the *threshold rejection score*).

<b>CasFinder system availability and citation</b>
---

The CasFinder system components listed in [Table S1](#) can be downloaded from <http://arep.med.harvard.edu/CasFinder> under the OSI-compliant MIT license (<http://opensource.org/licenses/MIT>).

Please cite the article:

Aach, J., Mali, P., Church, G.M. (2014) CasFinder: Flexible algorithm for identifying specific Cas9 targets in genomes. bioRxiv (doi)

if you use, copy, or employ code from the CasFinder system.

Please note also that the programs are made available as-is and that no support is provided. Some initial customization will be required in order to use the program (see section [CasFinder system installation and customization](#) below). The CasFinder system has been developed, tested, and used on 64-bit Debian GNU/Linux 6.0 ("squeeze") systems using amd64 processors using perl v5.10.1. We cannot give assurance that the programs will run as described here on other systems.

## CasFinder system installation and customization

Installing the CasFinder system involves four steps: (1) install program components, (2) prepare genome files, (3) edit the system configuration files, (4) test the installation. It is recommended that users read the section on [CasFinder system performance considerations](#) when installing the software.

### 1. Installing CasFinder program components

To install the CasFinder system, download the system component files from the URL listed in [CasFinder system availability](#), and put them all in a single directory. The CasValue and CasFinder programs should always be executed from this directory to ensure that they can find and use the system configuration files. Please note that (as mentioned in that section), the system has been developed, tested, and used on 64-bit Debian GNU/Linux 6.0 ("squeeze") systems using amd64 processors using perl v5.10.1. We cannot give assurance that the programs will run as described here on other systems.

The CasFinder and CasValue programs are designed to be run from a command line prompt using the *perl* command. To test that the programs may be run from the perl on your system, the following commands may be used. These commands display program help messages that describe program parameters and usages.

```
> perl CasFinder.pl -h
```

```
> perl CasValue.pl -h
```

If these commands do not work, contact your systems administrator to find out how to get them to run on the perl available on your system.

### 2. Preparing genome files needed for CasFinder operation

Every execution of the CasFinder and/or CasValue programs requires that you specify the genome in which you wish to evaluate Cas9 sites for specificity. In order for the programs to perform this evaluation, the following files must be available for each genome you plan to use them with:

1. Separate FASTA files for each chromosome or scaffold in the genome you wish to analyze. ***These files should all be repeatmasked with repeatmasked regions in lower case, and they should all be located in a common directory.*** For widely analyzed genomes such as human and mouse, files meeting these specifications can be downloaded from the UCSC Genome Browser [1,2,13] or NCBI websites. Instructions given in the template CASFINDER\_CONFIG.txt file that users must customize below (see [3. Editing the CasFinder system configuration files](#)) will direct users to set up a directory containing chromosome FASTA files for the UCSC hg19 human genome needed for the test in [4. Testing the installation](#).
2. A bowtie [5] index to the genome, either downloaded from a source such as <http://bowtie-bio.sourceforge.net/index.shtml>, or prepared using the *bowtie-build* program as instructed on that site. *The bowtie index for the genome should be built from or consistent with the same sequences that were collected in step 1 above.*

The files that you collect or generate in 1 and 2 will be declared in the CASFINDER\_CONFIG configuration file in [3. Editing the CasFinder system configuration files](#) below.

Please note that the FASTA chromosome files obtained for CasFinder in 1 above can be a *proper subset* of the sequences included in the bowtie index. Only FASTA files for that subset of sequences that you will wish to analyze for Cas9 sites and include in searches for possible off-targets need be included in 1. Chromosome and scaffold sequences downloaded from sequence databases such as those mentioned above often include files for variant sequences (such as alternative HLA regions in human), mitochondrial sequences, or unplaced sequences, that you may *not* want to search or check in Cas9 target searches. These can be included in the bowtie index, but the CasFinder system will ignore them if they are not in the CASFINDER\_CONFIG file. Further information on this point is given in [3. Edit the CasFinder system configuration files](#) below.

### 3. Editing the CasFinder system configuration files

The CasFinder system features two user-editable text configuration files (see [Table S1](#)): The CASFINDER\_CONFIG file specifies program and data resources needed by the CasValue and CasFinder programs to analyze genomes, and the CASFINDER\_CAS9\_CONFIG defines the characteristics of different Cas9 proteins. Of these two files, the first, CASFINDER\_CONFIG, *must* be edited as part of the process of installing the system in order for the system programs to operate. By contrast, the CASFINDER\_CAS9\_CONFIG file specifies Cas9 properties cited in the CRISPR literature and should not need to be updated during system installation. However, in the course of using the CasFinder system, users may encounter needs to change or add Cas9 definitions, so instructions for updating this file are also provided here.

#### 3.a Updating the CASFINDER\_CONFIG file

As just noted, the CASFINDER\_CONFIG file *must* be updated during CasFinder system installation. The copy of CASFINDER\_CONFIG.txt distributed with the system is a *template file only* that contains extensive comments on how to modify the file. These comments direct the user to customize the file in the way required to perform the test described in the section [4. Testing the installation](#) below. To further assist users with this step, an excerpt of the CASFINDER\_CONFIG used at the authors' computer installation is included as [Figure S3](#). This can be used as a model for filling out the required fields.

The lines of the CASFINDER\_CONFIG file are either

- Comments ignored by the system, indicated by the presence of a # as the first non-whitespace character in the line, *or*
- Declarations of a program or data resource needed or used by the system. Lines of this form contain two white-space separated character strings, the first of which specifies a keyword or resource name that is recognized by the system, and the second of which specifies its value.

```

# CASFINDER_CONFIG file
#
temp_file_prefix      /tmp/PROGRAM.TIME
#### EXECUTABLES
bowtie_executable     /opt/bowtie/bowtie
casvalue_program      /home/jda2/CRISPR/CasValue_v2.pl
#### GENOMES
#### hg19 human genome based on UCSC Genome Browser files
genome hg19
bowtie_index          /opt/bowtie_indexes/hg19
chromosomes           /hms/scratch1/shared_databases/igenome/Homo_sapiens/UCSC/hg19/Sequence/Chromosomes
chr1      chr1.fa
chr2      chr2.fa
chr3      chr3.fa
chr4      chr4.fa
chr5      chr5.fa
chr6      chr6.fa
chr7      chr7.fa
chr8      chr8.fa
chr9      chr9.fa
chr10     chr10.fa
chr11     chr11.fa
chr12     chr12.fa
chr13     chr13.fa
chr14     chr14.fa
chr15     chr15.fa
chr16     chr16.fa
chr17     chr17.fa
chr18     chr18.fa
chr19     chr19.fa
chr20     chr20.fa
chr21     chr21.fa
chr22     chr22.fa
chrX      chrX.fa
chrY      chrY.fa
end        chromosomes
#### mm10 mouse genome based on UCSC Genome Browser files
genome mm10
bowtie_index      /home/jda2/CRISPR/mouse_CRISPR/mm10
chromosomes       /hms/scratch1/shared_databases/igenome/Mus_musculus/UCSC/mm10/Sequence/Chromosomes
chr1      chr1.fa
chr2      chr2.fa

```

**Figure S3:** Excerpt of CASFINDER\_CONFIG file used at the authors' computer installation exemplifying formatted system statements at the top (*temp\_file\_prefix*, *bowtie\_executable*, and *casvalue\_program*), and a complete genome definition for the hg19 human genome downloaded from the UCSC Genome Browser [1,2]. Only chromosomes 1-22, X and Y from the genome are defined (see note 1 in this section). The start of a second genome definition (for mouse) is seen at the bottom.

The following keyword and resource name specifications must be updated at system installation time:

*temp\_file\_prefix*: This must specify a fully qualified filename prefix for temporary files that will be used by the system. It is important to specify a prefix that will be distinct for concurrent executions of the CasFinder system. To assist in the construction of such prefixes, the two words PROGRAM and TIME may be included in *temp\_file\_prefix* specification. The word PROGRAM is converted to the program name of the CasFinderConfigFile perl package that interfaces with the CASFINDER\_CONFIG file (see [Table S1](#)). Including PROGRAM in *temp\_file\_prefix* helps ensure that the temporary files used by the CasFinder system will have names distinct from those of temporary files used by other systems running in the user's computer installation. The word TIME is converted to a system timestamp that reflects the time of the current execution of the CasFinder system. Using TIME helps ensure that concurrent executions of the CasFinder system itself will use distinct temporary files. When used, the words PROGRAM and TIME must be provided in upper case. The specification of the *temp\_file\_prefix*

in the CASFINDER\_CONFIG file distributed with the system is /tmp/PROGRAM.TIME, which specifies that temporary files will be in the /tmp directory. Although this is not required by the system it is recommended because, while temporary files are normally deleted automatically by the system after use, large temporary files may be left behind in cases where a CasFinder execution ends in error or is cancelled by the user. Specifying that temporary files be in a /tmp directory that is regularly monitored and cleaned up will help ensure that such files do not accumulate.

*bowtie\_executable*: This must specify a fully qualified filename for a copy of the bowtie executable in the user installation. The bowtie program [5] can be downloaded from <http://bowtie-bio.sourceforge.net/index.shtml>.

*casvalue\_program*: This must specify the fully qualified filename of the CasValue perl program (see [Table S1](#)). The CasFinder program employs this as the default CasValue program file name for when CasFinder calls CasValue. This default can be overridden by CasFinder program options when users wish to test or run an alternate version of the CasValue program (see [CasFinder program options](#) below).

*Genome-specific resources that must be updated in CASFINDER\_CONFIG at system installation time:*

For each genome that the system is to process, groups of statements comprising a single *genome* statement, a single *bowtie\_index* statement, a single *chromosome* statement, and then a statement for each chromosome, must be placed *in this order* in the CASFINDER\_CONFIG file.

*genome*: This must specify a symbolic name for the genome, and must be the first statement in the group. CasFinder system programs may be directed to use any genome defined in the CASFINDER\_CONFIG file by specifying this symbolic name as the value of the *-g* program option.

*bowtie\_index*: This must specify the bowtie index that has been built for the genome (see [2. Preparing genome files needed for CasFinder operation](#) above). *bowtie\_index* should be specified in the form that bowtie expects to see in bowtie command line executions (see <http://bowtie-bio.sourceforge.net/index.shtml>), and should be directory-qualified as required for bowtie execution.

*chromosomes*: The *chromosomes* statement heads a set of CASFINDER\_CONFIG statements that provide the file names of the chromosome FASTA files described above in the section [2. Preparing genome files needed for CasFinder operation](#). The value specified in the *chromosomes* statement itself must be the fully-qualified name of the directory in which these FASTA files reside. Following the *chromosomes* line, a separate line must be provided that identifies the file name within this directory for each individual chromosome FASTA file in the genome that the CasFinder or CasValue programs may need to analyze. The format of these lines is:

*chromosome\_symbolic\_name*      *chromosome\_fasta\_file\_name*

Following all the lines identifying the individual chromosome files for the genome, the section must be terminated with the statement:

*end chromosomes*

Users should be aware of two considerations regarding chromosome symbolic names:

1. The symbolic names assigned to each chromosome in the CASFINDER\_CONFIG file must match the names of the chromosomes that were incorporated into the bowtie index for the genome. This is because after the CasValue program invokes bowtie to find potential off-targets in the genome for candidate target Cas9 sites, it looks at the sequence names given in the bowtie output with each bowtie match and only considers those names that match the chromosome symbolic names defined in CASFINDER\_CONFIG. This is why the bowtie index described in the section [2. Preparing genome files needed for CasFinder operation](#) may contain additional or miscellaneous genome sequences of no interest to the user that CasValue and CasFinder will ignore.
2. The order of the chromosomes given in the chromosome section is also the order used by CasFinder and CasValue when they extract sequences from the chromosome files in the course of their internal operations. This affects some program output, such as the CasValue *site\_detail* files (see [CasValue output files](#) below).

### 3.b Updating the CASFINDER\_CAS9\_CONFIG file

As noted above, it should not be necessary for users to edit this file at installation time. However, there may be occasions when users will want to add or adjust Cas9 definitions, so instructions for updating this file are given here.

As with the CASFINDER\_CONFIG file, two types of lines may be found in the CASFINDER\_CAS9\_CONFIG file:

- Comment lines, again indicated by # as the first non-whitespace character, *or*
- Definitions of a Cas9 target site for a particular Cas9.

Definitions of Cas9 target sites are given by lines containing six whitespace-separated values in a specific order. In order, the six fields are:

1. The symbolic name by which a particular Cas9 will be known to the CasFinder system.
2. The length of targeting sequence for this Cas9.
3. The starting position of the seed region within this targeting sequence (where the position of the first base of the targeting sequence is 1).
4. The ending position of the seed region within the targeting sequence.
5. A comma-separated list of specifications of principal PAMs for this Cas9.
6. A comma-separated list of specification of secondary PAMs for this Cas9. (If no secondary PAMs are recognized for this Cas9, this field need not be provided or can be left blank.)

Regarding principal and secondary PAMs: CasFinder only searches for candidate Cas9 targets that end in principal PAM sequences, while CasValue looks for Cas9 footprints matching candidate Cas9 targets bearing either principal or secondary PAMs in evaluating off-targets. It is to be recalled from the main article that secondary PAMs (item 6) are those that Cas9 recognizes with reduced activity, and that CasValue adds a user-specifiable cost to the score of potential off-targets to a candidate site that match secondary PAM motifs.

```
# (excerpt from CASFINDER_CAS9_CONFIG.txt...)
##### Cas9 CONFIGS: size default_seed_start(first pos=1) default_seed_stop primary...
SP      20      8      20      NGG      NAG
##### ST1 and NM from Esvelt et.al. (2013) Nat. Methods 10:1116.
ST1     20      8      20      NNRGAA,NNAGGA      NNANAA,NNGGGA
NM      20      8      20      NNNNGANN,NNNNGTTN,NNNNGNNT  NNNNGTNN,NNNNGNTN
```

**Figure S4:** Excerpt of distributed CASFINDER\_CAS9\_CONFIG.txt files showing definitions of three Cas9s. See text for details.

PAM specifications are given as sequences that may contain standard ambiguous base codes. Where multiple PAM specifications are given *all PAM sequences must be of the same length*. If the Cas9 recognizes PAMs of different lengths, this constraint can be met by padding the shorter PAMs with N wildcard base codes.

The CASFINDER\_CAS9\_CONFIG file distributed with the system contains definitions for the Cas9s from *S. pyogenes* (SP), *S. thermophilus* (ST1), and *N. meningitides* (NM) [11]. To assist the user in understanding how these definitions are structured, the definition statements copied from the distributed file are copied into [Figure S4](#) and discussed here:

As a convention, the symbolic names given for these Cas9s are simply the species signifiers SP, ST1, and NM. All of these Cas9s are assumed to have 20bp long targeting sequences whose last 13bp comprises their seed regions (i.e, both have seed starting position 8 and seed ending position 20). SP Cas9 has a single principal and a single secondary PAM: NGG and NAG, respectively, while ST1 and NM each have multiple principal and secondary PAMs given as comma-separated sequences. The use of N in all PAMs, and R in the first principal PAM for ST1, illustrate the use of ambiguous base codes in these PAM specifications. For each Cas9, all PAM specifications are the same length, and examples may be seen in NM where terminal Ns are used to pad shorter PAM sequences to meet this constraint.

It may also be noted that PAM specifications may be such that a sequence may match *both* a principal *and* a secondary PAM: e.g., the sequence CCCCGTTC matches both the NM principal PAM NNNNGTTN, and the NM secondary PAM NNNNGTNN. In any such cases, CasFinder and CasValue preferentially consider the sequence to match the principal vs. the secondary PAM.

#### 4. Testing the installation

When you downloaded the CasFinder system program and configuration file components from the URL specified in the section [CasFinder system installation and customization](#), you also received the file CasFinder\_installation\_test.fa and a folder named installation\_test\_results. The installation test will direct CasFinder to find *S. pyogenes* (SP) Cas9 targets in the sequences specified in CasFinder\_installation\_test.fa that have passed specificity checks for off-targets in the human genome. The installation\_test\_results contains output files from a successful execution of the CasFinder system on these same sequences that can then be compared with the output files obtained from the test.

CasFinder\_installation\_test.fa is a CasFinder input file that specifies two sequences:

1. the first exon of the TP53 gene isoform identified as ‘canonical’ in the UCSC Genome Browser [1,2,13], whose actual sequence CasFinder will extract from the chromosome files that were set up in section [3.a Updating the CASFINDER\\_CONFIG file](#). By means of the ‘accept=1’ keyword in

the FASTA header, the CasFinder system will treat this sequence as endogenous to the hg19 human genome in its search for potential Cas9 off-targets.

2. a sequence for an EGFP gene, given directly in the file. By means of the 'accept=0' keyword, the CasFinder system will treat this sequence as exogenous to the hg19 human genome.

For details on the keywords and options given in the FASTA headers, see the section [CasFinder input file specifications](#) below.

To conduct the test, go into the directory in which the CasFinder system program components have been installed, and execute the command:

```
> perl CasFinder.pl -i CasFinder_installation_test.fa -o CasFinder_installation_test
```

If the system has been installed correctly, it may take a few minutes for this command to complete, after which the system will return you to the command prompt. The receipt of any system of CasFinder error messages will indicate a problem with the system installation.

If the system has completed normally, you will find that four files have been created:

- CasFinder\_installation\_test.data.txt
- CasFinder\_installation\_test.stat.txt
- CasFinder\_installation\_test.info.txt
- CasFinder\_installation\_test.CasValue\_v2.pl.info.txt

The first two files, CasFinder\_installation\_test.data.txt and CasFinder\_installation\_test.stat.txt comprise the primary output of the system. The 'data' file will contain the SP Cas9 targets in the input sequences that have passed the system's checks for potential off-targets in the hg19 human genome, and the 'stat' file will list how many such targets were found for each input sequence. (Note that the command above did not explicitly specify SP Cas9 or the hg19 genome, as these are system defaults.) Proper installation and program operation will be confirmed if these files are identical with the files with the corresponding names in the `installation_test_results` directory. Identity may be confirmed with a *cmp* or *diff* command.

The other two files contain processing messages put out by the CasFinder and CasValue programs. While their content should match the corresponding files in the `installation_test_results` directory, they will not be byte-identical because they contain timestamps and installation-specific information.

Details on the contents and structure of all four of these output files can be found in the section [CasValue output files](#) below.

## General features of CasFinder and CasValue operation

### ***Program option attributes and coordination***

As noted in the text of the main article, the CasFinder system provides two programs: CasFinder, which is used to find Cas9 targets in user-specified sequence, and CasValue, which is used to evaluate the specificity of Cas9 targets in a genome. As also noted in the article, CasValue, while called automatically by CasFinder, can also be run independently of CasFinder. This allows CasValue to be used to evaluate

Cas9 target sites found by other systems or algorithms. A second feature of the CasFinder system, noted above in the section [Overview of the CasFinder system of programs](#), is that information that defines the characteristics of Cas9 proteins, and required resources like genome sequences, is maintained in user-editable configuration files outside of but shared by the CasFinder and CasValue programs themselves. This enhances the flexibility of the system by making it easy for users to add or change these kinds of information, and also assures that any such changes are picked up simultaneously and used consistently by the two programs. However, further flexibility is provided by a large assortment of program options. While some of these options control processing logic that is specific to CasFinder and CasValue individually, others control processing that must be coordinated between the programs. These coordinating options provide a second level of information sharing between CasFinder and CasValue beyond the common configuration files. Some options control coordinated processing that is completely independent of the configuration files, while others coordinate usages of or overrides to information in these files. To help convey how to use these program options correctly, each option used by each program is described by three attributes: *Default*, *Config*, and *Usage*, which are briefly explained here along with letter code abbreviations that are used in [Tables S2](#) and [S3](#).

*Default* attribute: This attribute describes where a program gets the value associated with a program option if it is not provided by the user *via* a command line invocation.

- *Required* (letter code = R): These options have no defaults either in the program or in a configuration file and so must be explicitly supplied by the user in the command line when invoking the program. An example is the `-i` parameter, which specifies the input file of sequences or sequence specifications that the program is to process.
- *Program* (P): These options have defaults defined in the program, so that, unlike *Required* options, the program does not require a value to be explicitly supplied when the program is invoked. An example is the CasFinder `-t` option (see [Table S2](#)). Of course, *Program* defaults may not always be appropriate for specific applications. Also, the *Program* defaults of some program options require related supporting information in a configuration file for proper operation of the default. For instance, the CasFinder `-c` option, which specifies the Cas9 the program is to process, has the *Program* default SP (*S. pyogenes*) but this can only be processed correctly if a valid definition of the SP Cas9 is in the CASFINDER\_CAS9\_CONFIG configuration file (see [Figure S4](#)). (This relationship to a configuration file is discussed in the section on *Config* attributes below.)
- *Switches* (S): These options are essentially *Program* options that effect a switch in program logic and do not require a command line value to be provided when specified during program invocation. An example is the `-R` option of CasFinder, which directs CasFinder to override its default action of ignoring candidate Cas9 target sequences in repeatmasked sections of sequence specified in its input file. In contrast to normal *Program* options like `-t`, which need to a value to be provided if used (e.g., in a command like `>perl CasFinder.pl -t 6`), the *Switch* option `-R` would be specified by `>perl CasFinder.pl -R` with only the `-R` option itself and no following value. Other than the `-R` option, there is only one other *Switch* program option in the CasFinder system: the `-D` option (see [Table S2](#) and [Table S3](#) below).
- *Forcing* (F): These are program options that cause the programs to override processing options specified with input sequences, see the section [Input sequence processing options](#) below. There are only two *Forcing* options in the CasFinder system, the CasFinder `-R` option and the CasValue `-A` option (see [Table S2](#) and [Table S3](#) below). The CasFinder `-R` option also has the *Switch* attribute.

- *None* (N): This code is primarily used to identify options that CasFinder receives and passes to CasValue, but does not use itself. Here CasFinder accepts the option but does not supply a default, and then CasValue will use its *own* default. An example is the `-x` parameter in CasFinder, which it does not use but passes to CasValue if provided. (This implies that the same option `-x` may have a *Default* attribute of *None* in CasFinder and a *Default* attribute of *Program* in CasValue.)

*Config* attribute: This attribute describes how the program option may override or have a required relationship to information in configuration files. There are three possible values:

- *Override* (letter code = O): These options, if provided, will cause the program to override specific values defined in the configuration files. An example is the `-ss` option of CasValue, which will cause CasValue to use the value provided with the option as the starting seed position in the targeting sequence of the Cas9 being processed, instead of the starting seed position provided in the CASFINDER\_CAS9\_CONFIG file (see [Table S1](#)) for that Cas9.
- *Specify* (S): These options specify values that must be defined in a configuration file for proper functioning of the option. An example is the `-c` option that specifies the Cas9 that the program will process. The value of the `-c` option must be the name of a Cas9 defined in the CASFINDER\_CAS9\_CONFIG configuration file (see [Figure S4](#)).
- *None* (N): These options have no direct relation to information in either of the configuration files.

*Usage* attribute: The *Usage* attribute conveys whether a program option is used by CasFinder or CasValue individually, or is used by both, and also controls the related question of whether the option, when specified for CasFinder, is passed through to CasValue. The values of the *Usage* attribute are described below, after which is given a description of the relationship between option *Usage* and the process of passing through parameters.

- *CasFinder* (letter code = F): These options provide information that is used exclusively by CasFinder.
- *CasValue* (V): These options provide information that is used exclusively by CasValue.
- *Both* (B): These are options that are used by both programs.

*The passing through of parameters:* When CasFinder invokes CasValue to evaluate candidate Cas9 targets, it calls the CasValue program using CasValue program options. However, CasFinder does not always itself use the options it passes to CasValue; some are simply accepted by CasFinder from its own command line for later passage to CasValue without CasFinder using them in its own processing.

Specifically, CasFinder options carrying the *Both* (B) attribute value are both used by CasFinder and passed to CasValue so that it, too, will use the same option. An example of a *Both* attribute is the `-c` program option, which specifies the Cas9 which is to be processed. CasFinder needs the `-c` option to know the Cas9 whose sites it must search for in the sequences specified in its input file. But when it calls CasValue to evaluate the specificity of the candidate sites it has found, it must pass the `-c` value on to CasValue so that CasValue, too, knows what Cas9 it should be considering in looking for genomic off-targets.

CasFinder options carrying the *CasValue* (V) attribute value are also passed through to CasValue, but CasFinder itself does not make any other use of them. The only reason CasFinder accepts these

options is so that users have a way to specify the corresponding CasValue settings, since the user makes no direct call to CasValue in these circumstances.

By contrast, CasFinder options carrying the *CasFinder* (F) attribute value are used *exclusively* by CasFinder and are not passed through to CasValue.

When CasValue is called directly from a command line instead of being invoked indirectly by CasFinder, it will expect to receive *Both* (B) and *CasValue* (V) options directly from the command line, or use its own internal *Program* or *Switch* defaults. Program options with the *CasFinder* (F) attribute are not recognized by CasValue, since it has no need of them.

Finally, a special provision, the `-CV` option, exists to allow users to pass parameters to CasValue when CasValue is called from CasFinder, for CasValue options that are not recognized by CasFinder. For instance, if the user calls CasFinder but wants CasValue to evaluate off-targets using exactly matching seeds vs. seeds with a single base mismatch, CasValue must receive the `-sm 0` option (see [CasValue program options](#)), but CasFinder does not recognize this option and will reject it. In this case, CasFinder can be called from the command line in the following way

```
>perl CasFinder.pl [CasFinder options] -CV "-sm 0"
```

In general, CasFinder will pass any string specified in the `-CV` option to CasValue when it calls the latter routine.

### ***Input sequence processing options***

In addition to program options and configuration file settings, which control general features of CasFinder and CasValue operation, both of these programs allow some processing options to be specified individually for each input sequence in their input files. For both programs, input files contain sequences or sequence specifications in a modified FASTA file format, and some processing options may be specified by special keywords in the FASTA headers. This arrangement allows the user to specify different processing that is appropriate for different input sequences directly, without having to break the input file into separate parts and process the parts separately using different program or configuration file options. An example is the *searchrepeat* instruction in CasFinder input files, which tells CasFinder whether or not it should look for candidate Cas9 targets in repeatmasked sequence. By default, CasFinder does not look for targets in such sequence, but this can be overridden by specifying *searchrepeat=Y* in the FASTA headers of input sequences for which this behavior is desired. Similarly the *accept* keyword can be specified in the FASTA headers of both CasFinder and CasValue input sequences to tell these programs how many instances of sequences that closely match the input sequence in the genome will be considered acceptable. Details on these input file options are given in the sections on input file specifications for both of these programs below.

The ability to specify processing options separately for each input sequence enhances the flexibility of CasFinder and CasValue, but there may be occasions where one wants to override all such input file specifications and process all input sequences the same way. For this reason, program options have been implemented that allow the user to override some input sequence processing options and process all the input sequences in a uniform way. These are identified as *Forcing* options (see the *Default* attribute above), and there are only two of them: the CasFinder `-R` program option, that instructs CasFinder to search for candidate Cas9 sequences in all of its input sequences regardless of whether or how *searchrepeat* is set individually, and the CasValue `-A` option, which overrides input sequence-specified *accept* numbers. Without *Forcing* options, users would be required to change the input

sequence specifications of every sequence in the input file should they want to ignore these specifications at a later time.

### CasFinder operation

CasFinder is designed to be executed from a linux operating system command line using the command

*perl CasFinder.pl [program options]*

Program options are described below. Help messages describing the options will be displayed if you use one of the commands:

*perl CasFinder.pl -h*  
or  
*perl CasFinder.pl -?*

Please note that these command forms above assume the name of the CasFinder program file specified in [Table S1](#).

#### **CasFinder program options**

Option	Attributes			Description
	Default	Config	Usage	
-A	F	N	V	Instructs CasFinder to pass the -A option to CasValue to override 'accept' numbers that may be specified in the input file CasFinder passes to CasValue. See the sections on <a href="#">Input sequence processing options</a> and <i>CasValue input files</i> for details.
-c	P	S	B	Cas9 for which CasFinder will search input file sequences for candidate target sites, and for which CasValue will be instructed to check for possible off-target sites. The Default is SP, which must therefore be defined in the CASFINDER_CAS9_CONFIG configuration file (see <a href="#">Figure S4</a> ).
-CASVALUE_PROGRAM	N	O	F	The program file that CasFinder will invoke to run the CasValue program. This parameter allows CasFinder to be directed to use alternate versions of the CasFinder program. The default value, CasValue_v2.pl (the program name specified in <a href="#">Table S1</a> ), is specified in the CASFINDER_CONFIG file.
-CONFIG	P	S	B	The name of the CASFINDER_CONFIG configuration file that will be used by CasFinder and CasValue. This parameter allows the user to specify alternative versions of this configuration file. The default is CASFINDER_CONFIG.txt (the file name specified in <a href="#">Table S1</a> ).
-CONFIG_CAS9	P	S	B	The name of the CASFINDER_CAS9_CONFIG configuration file that will be used by CasFinder and CasValue. This parameter allows the user to specify alternative versions of this configuration file. The

				default is CASFINDER_CAS9_CONFIG.txt (the file name specified in <a href="#">Table S1</a> ).
-CV	N	N	V	A special parameter used to pass program options to CasValue, or to override CasValue program defaults, for options that are not passed through to CasValue by CasFinder. See the section on <a href="#">The passing through of parameters</a> in the section Program option attributes and coordination above.
-D	N	N	V	Instructs CasFinder to pass the -D option to CasValue to generate a CasValue <i>site_detail</i> output file. See the section on <a href="#">CasValue output files</a> for details.
-g	P	S	B	Genome from which input sequences specified with the *EXTRACT* keyword will be searched for candidate Cas9 sites, and for which CasValue will be instructed to search for possible off-targets. The default is hg19, which must be defined in the CASFINDER_CONFIG configuration file for proper function (see <a href="#">Figure S3</a> ).
-i	R	N	F	Input sequence file of FASTA sequence specifications identifying those sequences to be searched for candidate Cas9 sites. See the section <a href="#">CasFinder input file specifications</a> for further information.
-k	N	N	V	The limit of the number of bowtie matches that bowtie will be instructed to return for each query when CasValue checks for off-targets for CasFinder-identified candidate Cas9 sites. See the main text of the article, and the section on <a href="#">CasValue program options</a> below, for details. If not provided, CasValue's own default will apply.
-o	R	N	N	Prefix for output files generated by CasFinder. See the section <a href="#">CasFinder output files</a> below for further information
-R	S F	N	F	Forces CasFinder to search for candidate Cas9 sites in repeatmasked regions of input sequences (indicated by lower case base codes), despite any input sequence-specified instructions to ignore repeatmasked sequence.
-t	P	N	F	T-string length. Cas9 guide RNAs that direct Cas9 activity to target sequences are usually expressed under Pol-III promoters that may interpret T-strings as transcription stop signals. In searching the input file for candidate Cas9 sites, CasFinder will filter out any site whose targeting sequence contains a string of Ts of length greater or equal to the value supplied for this parameter. The default value is 5.
-x	N	N	V	The score threshold at or below which a match to a candidate Cas9 targeting sequence will be considered by CasValue as applying against an input sequence's 'accept' number in CasValue's evaluation of CasFinder-identified candidate Cas9 sites. See the main text of the article, and the section on <a href="#">CasValue program options below</a> , for additional details. If not provided, CasValue's own default will apply.

**Table S2:** CasFinder program options, their *Attributes* (see text), and descriptions. Options are given in alphabetical order without regard to case. Required options for which values *must* be supplied by the user at program invocation are **highlighted**. Abbreviations for program option *Attributes* are as given in the section [Program option attributes and coordination](#) above.

### ***CasFinder input file specifications***

CasFinder accepts a modified form of FASTA input file, where the differences from ordinary FASTA format are:

- i. *Special keywords* may be optionally provided in the FASTA headers that will control aspects of CasFinder processing. In general, the format of a CasFinder input file FASTA header is:

```
>FASTA-ID-name *EXTRACT* range=seqname:seqstart-seqend refpoint=seq_location reorient=(+|-)
accept=(\d+) searchrepeat=(Y|N)
```

In this notation, items in regular font such as ‘\*EXTRACT\*’ and ‘range=’ must be coded as indicated when these options are used (although they are case-insensitive), items in *italics* represent values specified by the user, and the signifiers (+|-), (\d+), and (Y|N) represent regular expression patterns. (Thus, for instance, ‘reorient=(+|-)’ means that either of ‘reorient=+’ or ‘reorient=-’ may be coded, while ‘accept=(\d+)’ means that ‘accept’ may be assigned any non-negative integer.) The use and processing of these keywords will be explained below. Other annotations present in the FASTA header will be ignored by CasFinder except for being reported in the [CasFinder output files](#) as “other annotations.”

- ii. Whereas in ordinary FASTA format, a FASTA header must be followed by user-provided sequence, this can be optionally avoided if CasFinder is told where it can find and extract the sequence itself. This is controlled by the presence of the ‘\*EXTRACT\*’ and ‘range’ keywords in the FASTA header. The presence of ‘\*EXTRACT\*’ may also affect the way CasFinder processes other FASTA header keywords.

The \*EXTRACT\* option has been provided to make it easy for users to find specific Cas9 target sites within regions endogenous to the genome specified by the CasFinder *-g* parameter. Since CasFinder already knows where the chromosome sequences of these genomes are through its CASFINDER\_CONFIG file (see [3. Editing the CasFinder system configuration files](#)), it does not need the user to explicitly provide these sequences – if, *via* the ‘range’ keyword, it is told where to find them. The same CasFinder input file can contain a mixture of both ordinary FASTA records comprising headers and ensuing user-provided sequence, and \*EXTRACT\* headers without ensuing sequence. Examples of both kinds of sequence specifications are given in the input file provided for the CasFinder installation test procedure (see section [4. Testing the installation](#)).

In the following, the use of each keyword in the FASTA header will be described. Since processing may vary depending on whether the FASTA header specifies ‘\*EXTRACT\*’, the case of \*EXTRACT\* records will be handled first, and ordinary FASTA records will be described afterwards.

#### *Use and processing of FASTA header keywords in \*EXTRACT\* records*

**FASTA-ID-name:** This assigns a name that is used to relate Cas9 target sites and statistics reported in the [CasFinder output files](#) to the CasFinder input file sequences from which they originated.

**‘range’:** In an \*EXTRACT\* FASTA header, the ‘range’ keyword specifies a sequence region to be searched for specific Cas9 sites. *seqname* must correspond to a chromosome name in the

CASFINDER\_CONFIG file for the *-g* genome (see [3. Editing the CasFinder system configuration files](#)), *seqstart* specifies the starting location of the region, and *seqend* specifies its end location. CasFinder assumes that chromosome coordinates are specified such that the first base in the chromosome has location 1. The presence of 'range' within an \*EXTRACT\* record causes CasFinder to extract the specified region from the chromosome files and analyze it for Cas9 target sites. It also causes CasFinder to report any specific Cas9 sites that it finds in the sequence in the coordinates given in the 'range' keyword.

'refpoint' and 'reorient': These keywords allow users to define an optional sequence reference point within the 'range'. 'refpoint', if provided, must indicate a location *seq\_location* between the *seqstart* and *seqend* locations given in the range (inclusive). 'reorient' specifies the strand on which the 'refpoint' is considered to be oriented. Provision of these keywords will cause CasFinder to report the *offsets of the 5' ends of all Cas9 target sites* it writes to its output file relative to the reference point, in addition to reporting the location of the Cas9 target in the coordinates of its 'range'. For instance, if an \*EXTRACT\* record defines a sequence 'range' that contains a gene of interest, a 'refpoint' *seq\_location* may be used to specify the transcription start site of the gene, and 'reorient' may be used to specify the orientation of the gene. The Cas9 site offsets reported by CasFinder will then allow the user to easily identify sites upstream of the gene (these offsets will have negative values), and to judge the closeness of the site to the transcription start site (these offsets will have small absolute values). If 'refpoint' is not provided, it is assumed that *seq\_location* is the *seqstart* of 'range', and if 'reorient' is not provided, it is assumed to be +. (Note that since reported 'refpoint' offsets locate the 5' ends of a Cas9 site, a Cas9 site at a specific location in a sequence will have a different offset if it is on the + strand of the sequence than it will if it is on the - strand. Also note that the 5' end location is not the same as the Cas9-induced cut location.)

'searchrepeat': The 'searchrepeat' keyword instructs CasFinder whether or not to process candidate Cas9 targets in sequence regions that are repeatmasked, where this is indicated by lower-case base codes. As described in the main article, evaluation of candidate Cas9 targets in repetitive sequence can have substantial impacts on CasValue performance, so that the default behavior of CasFinder is to ignore repetitive sequence. However, there may be cases where users have a need to search for specific Cas9 sites within sequence that is repetitive, in which case 'searchrepeat=Y' may be specified for the sequence. Note that the CasFinder -R program option will force CasFinder to search for specific Cas9 sites in *all* of its input sequences.

'accept': The 'accept' number specifies how many matches a candidate Cas9 site found by CasFinder may be allowed to have in the genome with scores less than a specified threshold in order for it to be considered a 'specific' Cas9 site. This FASTA input specification is not used directly by CasFinder but is passed to CasValue as the 'accept' number to be used for every candidate Cas9 target sequence within the CasFinder input sequence (see [CasValue input file specifications](#)). For CasFinder input file \*EXTRACT\* records, CasFinder assumes a default of 1. This value is appropriate for finding specific sites within endogenous sequences because CasValue will automatically find at least one match with score 0 (i.e., the candidate target sequence itself), so that an off-target will only be indicated if at least one other genome location matching the target are found.

*Use and processing of FASTA header keywords in ordinary FASTA records (with no \*EXTRACT\* keyword)*

Where *\*EXTRACT\** is not specified, the user must provide the sequence to be analyzed by CasFinder after the FASTA header in the usual fashion. In this case, the special FASTA header keywords recognized by CasFinder are processed as follows:

*FASTA-ID-name* is processed the same way as it is for *\*EXTRACT\** FASTA records.

*'range'*, if provided, is treated as a comment by CasFinder and simply reported as an “other annotation” in the [CasFinder output files](#). Even if *'range'* is used to indicate a region of the *-g* genome, CasFinder does not assume that the provided input sequence has any relationship to this region, and neither extracts this region from a *-g* chromosome nor compares the input sequence against this region. Similarly, the *'range'* is not used to define the coordinates of any Cas9 site reported in the [CasFinder output files](#). Instead, locations of reported Cas9 target sites are treated as if they had a *'range' seqname* of *'FASTA'*, a *'range' seqstart* of 1, and a *'range' seqend* of *L*, where *L* is the length of provided FASTA sequence. Examples of output generated from such input sequences can be found in the output file provided with the CasFinder installation test procedure (see section [4. Testing the installation](#)).

*'refpoint'* and *'reforient'*: These keywords may be used in the same way as they are for *\*EXTRACT\** sequences, except that the *'refpoint'* must indicate a position between 1 and *L*, where *L* is defined above. If *'refpoint'* is not specified, a value of 1 is assumed.

*'searchrepeat'* is processed exactly as it is for *\*EXTRACT\** records. As in that case, repeatmasked sequence must be indicated by the use of lower-case base codes, although here these lower-case sequences must be present in the user-provided FASTA sequence vs. a *-g* chromosome.

*'accept'* values are treated exactly as they are for *\*EXTRACT\** records, except that here a default of 0 is assumed if *'accept'* is not explicitly specified (vs. 1 in the case of *\*EXTRACT\** records). The 0 *'accept'* default is appropriate under the assumption that, if user-provided FASTA sequence is being searched for Cas9 sites that have no off-targets in the *-g* genome, this is because this represents *exogenous* sequence being introduced into cells of the *-g* organism: Here, unlike the case of *\*EXTRACT\** sequences, it should not be expected that CasValue will automatically detect the candidate target site when looking for potential off-targets in the *-g* genome, so that *'accept'* does not have to be set to 1 to ignore this detection. Note, however, that this assumption may not be appropriate if the user-provided FASTA sequence is a close sequence variant of a region within the *-g* reference genome (such as a reference sequence modified by a SNP). Also note that CasValue's search *only* considers the *-g* genome, so that the presence of potential off-targets within the user-provided FASTA sequence itself (or any other ordinary FASTA sequence in the CasFinder input file) will not be checked.

### **CasFinder output files**

CasFinder generates four output files per execution, an *info file*, a *data file*, a *stat file*, and a *CasValue into* file, and will generate an additional *site\_detail* file if the *-D* program option was specified. The names of all output files begin with the prefix specified by CasFinder's *-o* parameter, so that the command

```
> perl CasFinder.pl -o OUTPUT-PREFIX [... other parameters]
```

will result in the four files OUTPUT-PREFIX.info.txt, OUTPUT-PREFIX.data.txt, OUTPUT-PREFIX.stat.txt, and OUTPUT-PREFIX.CasValue\_v2.pl.info.txt, and, possibly, a fifth file OUTPUT-PREFIX.CasValue\_v2.pl.site\_detail.txt. These are described in turn:

#### OUTPUT-PREFIX.info.txt

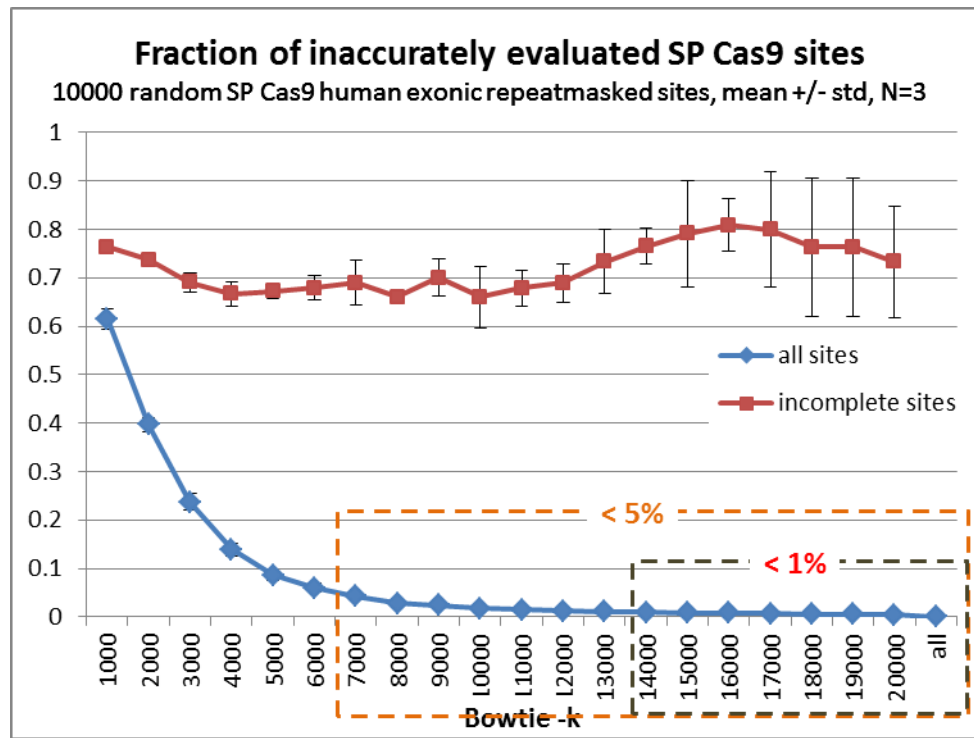
This is a text file containing informatory, error, and status messages about the program options and processing performed during the execution of CasFinder that generated the other files.

#### OUTPUT-PREFIX.data.txt

This is CasFinder's main data output and comprises a tab-delimited table each row of which represents a Cas9 site found in an input sequence that passed the specificity constraints in effect during the program execution (see text of main article and the section on [CasFinder program options](#) above). Cas9 sites in the data file are grouped together by input sequence, where the input sequences presented in the data file are in the same order as they were in the input file. Individual sites in any input sequence group are not presented in any particular order, but the information provided with each site allows users to sort them in many ways, e.g., by proximity to user-specified reference points, by absolute sequence locations, or by specificity (i.e., *threshold rejection score*). The columns of the data file are now described.

1. *target\_number*: The ordinal position of the Cas9 site described in the current line in the output data set.
2. *sequence\_number*: The ordinal position of the input sequence in which the current Cas9 site was found in the input sequence data file.
3. *sequence\_id*: The FASTA id of the input sequence in which the current Cas9 site was found.
4. *sequence\_range*: The sequence location of the input sequence as taken from the range specification in the input sequence's FASTA header (see the section on [CasFinder input file specifications](#) for details).
5. *sequence\_refpoint*: The user-specified reference point for Cas9 site locations in the input sequence, as specified by the *refpoint* specification in the input sequence's FASTA header (see the section on [CasFinder input file specifications](#) for details).
6. *sequence\_acceptnum*: The 'accept' number specified or assumed for the input sequence as taken from the input sequence's FASTA header or determined by CasFinder or CasValue defaults (see text of the main article and the section on [CasFinder input file specifications](#) for details).
7. *sequence\_other\_annotation*: Annotations that were in the FASTA header of the input sequence other than the ones described in the section on [CasFinder input file specifications](#) above.
8. *target\_threshold\_rejection\_score*: The CasValue threshold rejection score for the current Cas9 site. As described in the text of the main article and in the section on [CasValue program operation](#) below, this may be used to rank the specificities of different Cas9 sites in the output file, with higher values representing more specificity than lower values.
9. *target\_all\_matches\_analyzed?*: As described in the text of the main article and in the section on [CasValue program operation](#) below, this is an indicator which, when set to 1, signifies that CasValue examined *all* potential off-targets for the current site in the -g genome, and thus that

this site was comprehensively evaluated within the limits of the CasValue algorithm. A 0 value for the indicator signifies that CasValue only examined a subset of potential off-targets for the site (of size determined by the CasFinder and CasValue  $-k$  parameters, see text of the main article, and the sections on [CasFinder program options](#) and [CasValue program options](#) for details). Sites for which this indicator has the value 0 may have off-targets in the genome that were undetected by CasValue that would have caused it to fail specificity checks and be eliminated from the output file were they seen. It is also possible that the site would have passed specificity checks if all potential off-targets were examined, but that the *threshold rejection score* would have been lowered by this examination. See [Figure S5](#) for an estimate of the fraction of inaccurately evaluated SP Cas9 sites at various values of  $-k$ .



**Figure S5:** Fraction of inaccurately accepted or scored SP Cas9 sites identified by CasValue at different bowtie  $-k$  values compared to same sites as scored by CasValue with bowtie  $-a$  (return all matches). Three sets of 10000 candidate SP Cas9 sites of the form  $N(20)NGG$  were randomly extracted from repeatmasked hg19 human genome exome sequence and evaluated by CasValue at the indicated values of bowtie  $-k$  or using bowtie  $-a$  (x-axis). Sites accepted by CasValue, and the *threshold rejection scores* (TRSes) of these sites were compared for each bowtie  $-k$  value against the bowtie  $-a$  results. The fraction of inaccurately evaluated bowtie  $-k$  sites was calculated as the fraction of sites that were either (i) accepted in the bowtie  $-k$  but rejected in the bowtie  $-a$  run, or (ii) accepted in both runs but having a TRS in the bowtie  $-k$  run that was different (higher) than that in the bowtie  $-a$  run. *Red line:* Fraction of inaccurately evaluated sites reported in bowtie  $-k$  runs for those sites in which *all\_matches\_analyzed?* was reported as 0. *Blue line:* Fraction for all sites reported in bowtie  $-k$  runs. The exome sequence file used in this analysis is the file generated in ref. [3] rather than the file generated in this study. Note that in the SP Cas9 exome analysis performed in this study (see [Generation of the all human and mouse exome catalogs of Cas9 sites](#)) we chose  $-k$  20000 to bring the fraction of incompletely analyzed sites down to  $< 1\%$ , not the fraction of inaccurately evaluated sites.

10. *target\_nonstandard\_basecode\_matches*: This value, if not 0, signifies that the potential off-targets examined by CasValue for the current site contained among them instances of non-standard or ambiguous base codes. Since CasValue scores off-targets by sequence mismatches and the matching or mismatching of non-standard base codes is indeterminate, a non-zero value in this column means that under some interpretations of the non-standard codes it is possible that this site would have been scored differently.
11. *target\_location*: The sequence location of the current Cas9 site relative to the coordinates given in the input sequence's *range* specification (see the section on [CasFinder input file specifications](#) for details).
12. *target\_strand*: The orientation of the Cas9 site within the *target\_location* (where the input sequence is always assumed to have a + orientation).
13. *target\_refpoint\_5p\_offset*: The offset of the 5' end of the Cas9 target site relative to the input sequence's *refpoint* specification. As described in the section on [CasFinder input file specifications](#), this provides a way for users to quickly identify sites that are upstream or downstream of a user-specified reference point (e.g., the 5' end of a gene), and to quickly sort the Cas9 sites found for an input sequence by their proximity to this reference point (where sorting should be by the *absolute value* of the offset). Note, however, that the *refpoint* offset specifies the location of the 5' end of the Cas9 target site, not by the position of the dsDNA cut made by the Cas9 at this target, and that two Cas9 target sites at the same location that have opposite orientations will have different 5' offsets. [NOTE: That the CasFinder system computes 5' target site offsets and not offsets to cut locations was a design decision that reflects the fact that cut locations and types are not known for all Cas9s, and that cut locations may have no meaning for Cas9s that have been engineered to be nuclease-dead or otherwise altered.]
14. *target\_sequence*: The sequence of the current Cas9 target, complete with its PAM site, in the orientation recognized by Cas9.

#### OUTPUT-PREFIX.stat.txt

This file provides the number of specific Cas9 sites found and reported in the *data file* for each input sequence in the input file. Note that an input sequence in which no specific Cas9 sites were found will be found in the *stat file* with a 0 site count but will not be found in the *data file* at all. Rows of the *stat file* correspond one-per-one with and in the same order as input sequences in the input file. The columns of this table are as follows.

1. *sequence\_number*: The ordinal position of the input sequence in the input sequence data file.
2. *sequence\_id*: The FASTA id of the input sequence.
3. *sequence\_range*: The sequence location of the input sequence as taken from the range specification in the input sequence's FASTA header (see the section on [CasFinder input file specifications](#) for details).
4. *number\_reported\_targets*: The number of Cas9 sites reported for the input sequence in the *data file*.

#### OUTPUT-PREFIX.CasValue v2.pl.info.txt

This file is a copy of the info file generated by CasFinder's invocation of CasValue for this execution of CasFinder. See [CasValue output files](#) for details.

#### OUTPUT-PREFIX.CasValue v2.pl.site\_detail.txt

This file is generated if the `-D` option is specified to CasFinder. *site\_detail* files are generated by CasValue and contain information on all genome matches found to all candidate Cas9 sites with scores  $\leq$  the CasValue `-x` parameter. This file supplements information in the CasFinder output data file by indicating where the potential off-targets may be in the genome for all candidate targets that were rejected by CasValue. See [CasValue output files](#) for details.

### CasValue operation

CasValue is invoked automatically by CasFinder when CasFinder is called to search for Cas9 target sites in user-specified input sequences, but CasValue may also be invoked directly from a command line to evaluate Cas9 target sites found by other methods. CasValue may be called directly using the command

```
perl CasValue_v2.pl [program options]
```

Note that the program name indicated here is the name of the current program version given above in [Table S1](#).

CasValue program options are described below. Help messages describing the options will be displayed if you use one of the commands:

```
perl CasValue_v2.pl -h
or
perl CasValue_v2.pl -?
```

#### **CasValue program options**

Option	Attributes			Description
	Default	Config	Usage	
-A	F	N	V	This option, if specified, forces the 'accept' number of all CasValue input file sequences to the value provided. If it is not specified, CasValue uses the 'accept' numbers specified in each sequence's FASTA header if provided, and otherwise defaults to 1 for *EXTRACT* FASTA sequences and 0 for ordinary FASTA sequences. See the section <a href="#">CasValue input file specifications</a> for further Information.
-asc	P	N	V	The cost added to the score of a potential off-target site if an alternate (i.e., secondary) PAM is detected at this site. The default is 1.

-c	P	S	B	Cas9 for which CasValue search for possible off-target sites for target sequences in its input file. If CasValue is called from CasFinder, -c is passed through to it; otherwise, if CasValue is invoked directly, its default is SP. Note that this default will only be meaningful if the SP Cas9 is defined in the CASFINDER_CAS9_CONFIG configuration file.
-CONFIG	P	S	B	The name of the CASFINDER_CONFIG configuration file that will be used by CasValue. This parameter allows the user to specify alternative versions of this configuration file. When CasValue is called from CasFinder and this option is specified to CasFinder, it is passed through to CasValue. Otherwise, the default is CASFINDER_CONFIG.txt (the file name specified in <a href="#">Table S1</a> ).
-CONFIG_CAS9	P	S	B	The name of the CASFINDER_CAS9_CONFIG configuration file that will be used by CasValue. This parameter allows the user to specify alternative versions of this configuration file. When CasValue is called from CasFinder and this option is specified to CasFinder, it is passed through to CasValue. Otherwise, the default is CASFINDER_CAS9_CONFIG.txt (the file name specified in <a href="#">Table S1</a> ).
-D	S	N	V	This option, if provided, causes CasValue to generate a 'site_detail' output file that gives the locations and scoring information for each potential off-target found for each input file sequence that has a score <= the CasValue -x parameter. See <a href="#">CasValue output files</a> for details.
-g	P	S	B	The genome for which CasValue is to search for possible off-targets to Cas9 target sequences in its input file. When CasValue is called by CasFinder, CasFinder passes its -g value to CasValue; otherwise the default is hg19. Note that this default will only be meaningful if the hg19 genome is defined in the CASFINDER_CONFIG configuration file.
-i	R	N	F	Input sequence file of FASTA sequences and input file processing options for Cas9 target sequences that CasValue is to evaluate for possible off-targets in the -g genome. See the section <a href="#">CasValue input file specifications</a> for further information. When CasValue is called by CasFinder, CasFinder formats and passes a temporary -i file to CasValue containing all of the candidate Cas9 target sequences it has found during its site search.
-k	P	N	V	The limit of the number of bowtie matches that bowtie will return for each query when CasValue checks for off-targets of the Cas9 target sequences specified in its input file. See the main text of the article, and the section on <a href="#">CasValue program options</a> below, for details. The value 0 controls use of the bowtie -a option, which returns <i>all</i> bowtie matches in the genome. If CasValue is called by CasFinder, and -k is specified to CasFinder, CasFinder passes this value to CasValue. Otherwise, the CasValue default is 0.
-nsc	P	N	V	The score cost applied to the number of mismatches in the non-seed region of a potential off-target to an input target sequence, in computing this component of the potential off-target score. The default is 1.

-o	R	N	N	Prefix for output files generated by CasValue. See the section <a href="#">CasFinder output files</a> below for further information. When CasValue is called by CasFinder, it passes an -o prefix to CasValue specifying a temporary file (using the CASFINDER_CONFIG <i>temp_file_prefix</i> , see section <a href="#">3. Editing the CasFinder system configuration files above</a> ) and then processes the resulting output files when CasValue has finished. These files are later deleted by CasFinder (although a copy of the CasValue <i>info</i> and, if specified, the CasValue <i>site_detail</i> , files are retained).
-se	P	C	V	The location of the end of the seed region in a potential off-target site to an input sequence for the Cas9 currently being processed. If not provided, this location is taken from the CONFIG_CAS9 configuration file (see <a href="#">Table S1</a> ). The -ss and -se options exist to give users a way of analyzing potential off-targets using under different assumptions of the seed location and length, without their having to alter the configuration file.
-sm	P	N	V	The number of seed region mismatches that will be considered in finding seed region matches to input target sequences in the specified genome. This value is used as the -v option value in CasValue's call to bowtie to locate these seeds. The default is 1.
-ss	P	C	V	The location of the start of the seed region in a potential off-target site to an input sequence for the Cas9 currently being processed. If not provided, this location is taken from the CONFIG_CAS9 configuration file (see <a href="#">Table S1</a> ). The -ss and -se options exist to give users a way of analyzing potential off-targets using under different assumptions of the seed location and length, without their having to alter the configuration file.
-ssc	P	N	V	The score cost applied to the number of mismatches in the seed region of a potential off-target to an input target sequence, in computing this component of the potential off-target score. The default is 1.
-x	P	N	V	The score threshold at or below which a match to a candidate Cas9 targeting sequence will be considered by CasValue as applying against an input sequence's 'accept' number. When CasValue is called by CasFinder, and the CasFinder -x parameter is specified, it passes this value to CasValue. Otherwise, the CasValue program default is 3. See the main text of the article, and the section on <a href="#">CasValue output files</a> below, for additional details.

**Table S3:** CasValue program options, their *Attributes* (see text), and descriptions. Options are given in alphabetical order without regard to case. Required options for which values *must* be supplied by the user at program invocation are **highlighted**. Abbreviations for program option *Attributes* are as given in the section [Program option attributes and coordination](#) above.

#### **CasValue input file specifications:**

CasValue accepts a FASTA input file in the following format:

1. The sequences in the input file are *candidate Cas9 targeting sequences only*. This means that
  - a. The length of the FASTA sequence is the length of the targeting sequence as given in the CASFINDER\_CONFIG\_CAS9 file (see [Figure S4](#)) for the Cas9 being processed.
  - b. The FASTA sequences contain targeting sequences only and *not* targeting sequences followed by PAM sequences
  - c. The sequences must be given in the + sense only.
2. The FASTA header must contain a FASTA ID and can optionally contain an 'accept' number for the sequence. All other annotations in the FASTA header are ignored. Thus, the format of CasValue FASTA header is

```
>FASTA-ID [accept=\d+] other annotations...
```

where the \d+ in the accept clause is a regular expression indicating that any non-negative integer may be specified, and the brackets around accept=\d+ phrase indicate that it is optional.

## NOTES

1. If no 'accept' number has been specified for an input sequence, and the -A program option has not been used, the value of 1 is assumed. However, if the -A option *has* been used, it overrides all input file accept number specifications and is used for every input sequence.
2. When CasFinder calls CasValue, it generates a temporary CasValue input file containing the target sequences for every candidate Cas9 sequence that it has found in its own input file with whatever accept number (if any) was specified for the CasFinder input sequence in which the candidate site was found, and accept=1 if no accept specification was provided. CasFinder generates artificial FASTA-IDs for each candidate sequence, and resolves these back to locations within its input sequences when it analyzes the resulting CasValue output.

For example, supposing that CasFinder is processing an hg19 input sequence

```
>seq1 *EXTRACT* range=chr13:108870409-108870816 accept=1 (other options)
```

and finds the candidate SP Cas9 site TGGTGGACTCCCGAGCTCATCGG that in this sequence (which happens to be at chr13:108870528-108870550 on the + strand). This candidate will be presented in the temporary CasValue input file in the form:

```
>R0-108870528 accept=1
TGGTGGACTCCCGAGCTCA
```

where R0-108870528 is a generated name. These names contain the relative sequence number of the input sequence seq1 in the CasFinder input sequence file (this is the 0 in R0), and the position of this target in the input sequence itself. Note that the 'accept' number of seq1 is carried forward into the FASTA header of the temporary CasValue input file, and also that the targeting sequence only and not the CGG PAM is presented in this file.

The reason that the sequences given to CasValue to evaluate are targeting sequences only, and do not contain PAMs, is that, conceptually, CasValue's job is to count how many locations in a genome might be bound by Cas9 complexed with a particular sgRNA. Because sgRNAs only specify targeting sequences, and any genome location that matches the sgRNA followed by any PAM is a potential binding site as far as Cas9 is concerned, no purpose would be served by providing CasValue any particular PAM (e.g., the PAM found after a candidate targeting sequence by CasFinder) along with the targeting sequence.

### **CasValue output files:**

CasValue generates two output files per execution, an *info file* and a *data file*, and optionally, a *site\_detail* file if the `-D` program option is specified. As with CasFinder, the names of all output files begin with the prefix specified by CasValue's `-o` parameter, so that the command

```
> perl CasValue_v2.pl -o OUTPUT-PREFIX [... other parameters]
```

will result in the (up to three) files OUTPUT-PREFIX.info.txt, OUTPUT-PREFIX.data.txt, , and OUTPUT-PREFIX.site\_detail.txt. These are described in turn:

#### OUTPUT-PREFIX.info.txt

This is a text file containing informatory, error, and status messages about the program options and processing performed during the execution of CasValue that generated the other files. If CasValue was called by CasFinder, a copy of this file is saved under CasFinder's `-o` prefix (see the section [CasFinder output files](#) above).

#### OUTPUT-PREFIX.data.txt

This is CasValue's main data output file, and comprises a tab-delimited table in which the rows represent all and only those CasValue input sequences that passed CasValue's specificity checks, in the order of the sequences given in the CasValue input file. See the text of the main article, and the section on [CasFinder program options](#) above, for information on how CasValue evaluates specificity. The columns of the *data* file are:

1. *input\_targetseq\_index*: The ordinal position of the input targeting sequence in the CasValue input file corresponding to the current *data* file row, where the first input sequence in the input CasValue file has an *input\_targetseq\_index* of 0.
2. *Input\_targetseq\_name*: The FASTA-ID of the input file target sequence corresponding to the current *data* file row.
3. *target\_threshold\_rejection\_score*: The lowest value to which the `-x` parameter value could be raised that would cause the input target sequence corresponding to the current *data* file row to be rejected. As noted in the main article and in the sections on [CasValue](#) and [CasFinder program options](#) above, when evaluating possible locations in the `-g` genome that might be binding sites for a targeting sequence, CasValue looks at all Cas9 footprints in the genome that match the 'seed' subsequence of the input targeting sequence up to `-sm` mismatches, and computes a score for the footprint based on the number of mismatches in the seed and non-seed portions of the footprint, and on the presence of a principal or secondary PAM, using the scoring costs `-ssc`, `-nsc`, and `-asc`. CasValue keeps track of the total number of footprints it finds for the input targeting sequence that have scores  $\leq -x$ . If no more than the input sequence's 'accept' number of footprints have been found with scores  $\leq -x$ , the target is considered accepted and is represented in the output file. However, by keeping track of the lowest *m* scores, for *m* = one more than the 'accept' number, CasValue can additionally identify the lowest score  $> -x$  for any footprint found in the genome. This value is reported as the *target\_threshold\_rejection\_score*. Broadly speaking, the *target\_threshold\_rejection\_score* represents the distance in sequence space to the closest match in the `-g` genome of an accepted

target. Accepted targets with larger values of this score can therefore be considered more specific than accepted targets with smaller values. However, although this score is the result of a systematic and comprehensive calculation, its interpretation as a specificity ranking is limited by the heuristics incorporated into the scoring mechanism, including the restriction of Cas9 footprints to locations in the genome with only up to  $-sm$  mismatches in the seed sequence, and the scoring costs assigned to footprint mismatches and alternate PAMs. See also the description of the *all\_matches\_analyzed* indicator below.

NOTE: Occasions arise in which *no* Cas9 footprint is found in the  $-g$  genome for an accepted targeting sequence that has a score  $> -x$ . In these cases the *target\_threshold\_rejection\_score* is given the value *Inf*, signifying that the targeting sequence would be accepted no matter how high the  $-x$  value is set. Situations that encourage generation of *Inf target\_threshold\_rejection\_scores* include small  $-g$  genomes, large seed sizes, and use of  $-sm\ 0$  (i.e., instructing CasValue to use bowtie for exact seed matches vs. seed matches with mismatches).

4. *all\_matches\_analyzed*: A value of 1 indicates that CasValue has looked at all Cas9 footprints in the  $-g$  genome in deciding to accept the current input targeting sequence, while a value of 0 indicates that it may only have looked at a proper subset of these footprints. As noted in the main article and in the sections on [CasValue](#) and [CasFinder program options](#) above, CasValue performance depends greatly on the number of matches in the  $-g$  genome returned by bowtie to the seed regions of its input targeting sequences, and this performance can be partly controlled by using the  $-k$  program option to limit the number of bowtie returns per query. If the number of bowtie returns for an input targeting sequences seed query is  $< -k$ , *all\_matches\_analyzed* is set to 1; otherwise it is set to 0.
5. *matches\_with\_nonstandard\_bases*: This is a count of the number off Cas9 footprints in the  $-g$  genome found for the input targeting sequence that contain ambiguous base codes. These are reported because the scores computed for such footprints are not fully determinate, as they depend on the precise interpretations of the bases at the positions in question.

When CasValue is called by CasFinder, CasFinder reports the *target\_threshold\_rejection\_score*, *all\_matches\_analyzed*, and *matches\_with\_nonstandard\_bases* values computed by CasValue for every candidate Cas9 target sequence that passes the CasValue specificity checks in its own (i.e., CasFinder's) output *data* file.

#### OUTPUT-PREFIX.site\_detail.txt

When CasValue is called without the  $-D$  option, it reports only those input targeting sequences in its output *data* file that have passed the prescribed specificity checks, but gives no information on input targeting sequences that have failed these checks. By implication, these rejected targeting sequences have potential off-targets in the  $-g$  genome, but the number and location of these potential off-targets are not reported. The  $-D$  option generates a *site\_detail* file that contains this information. Specifically, the *site\_detail* file contains scoring and location information on each Cas9 footprint for each input targeting sequence that score  $\leq -x$ .

The *site\_detail* file is produced in a slightly different format when CasValue is called from CasFinder compared to when CasValue is called independently. When CasValue is called independently, the columns of the *site\_detail* file are:

1. *target\_index*: The ordinal position of the input targeting sequence in the CasValue input file corresponding to the current *site\_detail* file row, where the first input sequence in the input CasValue file has a *target\_index* of 0.
2. *target\_name*: The FASTA-ID of the CasValue input file target sequence corresponding to the current *site\_detail* file row.
3. *site\_location*: The location in the  $-g$  genome of the Cas9 footprint that has been found to match the input CasValue targeting sequence with score  $\leq -x$ . The location is given in the form *chr:startloc-endloc:strand*, where *chr* is the chromosome, *startloc* and *endloc* are the starting and ending locations of the Cas9 footprint, and *strand* is + or  $-$ .
4. *site\_score*: The score computed for the Cas9 footprint. As noted, this will always be  $\leq -x$ .
5. *total\_mismatches*: The total number of mismatches between the Cas9 footprint's targeting sequence and the input targeting sequence.
6. *alternate\_pam?*: 0 if the Cas9 footprint contains a principal PAM, and 1 if it contains an alternative PAM.
7. *seed\_mismatches*: The number of mismatches between the Cas9 footprint's seed region and the seed region of the input targeting sequence.
8. *nonseed\_mismatches*: The number of mismatches between the non-seed region of the Cas9 footprint and the non-seed region of the input targeting sequence.
9. *non\_acgt\_basecode?*: 0 if the Cas9 footprint contains only standard base codes (A, C, G, or T), 1 otherwise.
10. *site\_sequence*: The Cas9 footprint sequence.

When CasValue is called by CasFinder with the  $-D$  option, the input file submitted to CasValue refers to candidate target sites with computer-generated artificial names, and these generated names appear in CasValue's *site\_detail* file. Therefore, as these names would be difficult for users to interpret, CasFinder rewrites CasValue's *site\_detail* file and replaces columns 1 and 2 above with information to CasFinder's own input file. Note that there are now three sequences involved in every *site\_detail* file row:

- A. The sequence in CasFinder's input file in which CasFinder located candidate Cas9 sites.
- B. The candidate Cas9 sites located in sequence A. (Recall that only some candidate Cas9 sites reported in the *site\_detail* file will have been accepted by CasValue, and only these will be reported in CasFinder's output *data* file.)
- C. A Cas9 footprint in the  $-g$  genome that matched sequence B and with a score  $\leq -x$ .

The following describes the columns that CasFinder puts into the *site\_detail* file to replace CasValue's columns 1 and 2:

1. *Input\_sequence\_number*: The ordinal position of the input sequence in the CasFinder input file (i.e., sequence A), whose candidate Cas9 site (sequence B) generated the current *site\_detail* row, where the first input sequence in the input CasFinder file has a *input\_sequence\_number* of 1.

2. *sequence\_id*: The FASTA-ID of the CasFinder input file target sequence (i.e., sequence A) corresponding to the current *site\_detail* file row.
3. *sequence\_range*: The *range* of the CasFinder input sequence (i.e., sequence A), as specified by the range keyword in the CasFinder input sequence's FASTA header (see the section *CasFinder input file specifications*).
4. *candidate\_target\_range*: The location of the candidate Cas9 target site (i.e., sequence B) in the CasFinder input sequence (i.e., sequence A).
5. *candidate\_target\_strand*: The strand of the candidate Cas9 target site (i.e., sequence B).
6. *candidate\_target\_sequence*: The sequence of the candidate Cas9 target site (i.e., sequence B).
- 7-14. The content of columns 3-10 of CasValue's *site\_detail* file. All of this information now pertains to a Cas9 footprint in the -g genome (i.e., sequence C) that is matched against sequence B.

NOTES regarding the *site\_detail* file:

- *Order of rows in the file*: The CasValue *site\_detail* file is generated during CasValue's analysis of bowtie matches in the -g genome to input targeting sequence seeds in the order in which they are evaluated, and CasFinder does not re-order the file generated by CasValue. This order reflects the order in which bowtie returned seed matches, and also the grouping together of matches by chromosome location, which is governed by the CASFINDER\_CONFIG file (see Note 2 in the section [3.a Updating the CASFINDER\\_CONFIG file](#) in [CasFinder system installation and customization](#), above). This order does not group together potential off-targets by candidate Cas9 target site, or candidate Cas9 sites by CasFinder input sequences. The user must sort the file to see these groupings.
- *Size of the file*: The *site\_detail* file can also be very large and usually contains one or more rows for every input sequence given to CasValue, and so for every candidate Cas9 sequence (whether or accepted or not) generated by CasFinder.

<b>Sensitivity, specificity, and FDR of specific Cas9 site detection: Preliminary assessment</b>
--

In principle, the ability of CasFinder and other Cas9 site evaluation algorithms (see [Comparison of the CasFinder system with other Cas9 target evaluation algorithms](#)) to detect Cas9 target sites that lack off-targets in a genome can be tested by assessing their performance on a so-called 'ground truth' list of targets that have been well-characterized for off-targets. However, at this time, only a short and non-representative list of targets has been characterized in this way, making such assessments problematic. Nevertheless, because estimation of the sensitivity and accuracy of Cas9 target selection algorithms is of intrinsic interest, we here present a calculation of CasFinder's sensitivity, specificity, and False Discovery Rate (FDR). Afterwards we discuss caveats concerning this estimation, discuss the reasons for our limited ability to assess these parameters, and suggest steps that may eventually lead to more robust assessments.

[Table S4](#) presents a list of 21 distinct sites in the human genome that were targeted by SP Cas9 in human cell lines with a precisely matching sgRNA, and for which numerous computationally identified similar sites in the genome were then assessed for off-target Cas9 activity [6,14,15]. Given that CasFinder promises to identify candidate Cas9 target sites with reduced potential for off-target activity, we consider any site characterized as *not having* an off-target that CasFinder evaluated and accepted to be

#	target site	location (hg19)	Name	off-target?	ref.	CasFinder accept?	In GECKO?
1	GGGTGGGGGAGTTTCTCCTGG	chr6:43737291-43737313	VEGFA site 1	1	[6]	NA	0
2	GACCCCTCCACCCGCTCCGG	chr6:43738556-43738578	VEGFA site 2	1	[6]	0	0
3	GGTGAGTGAGTGTGCGTGTGG	chr6:43737454-43737476	VEGFA site 3	1	[6]	NA	0
4	GAGTCCGAGCAGAAGAAGAAGG	chr2:73160982-73161004	EMX1	1	[6]	0	0
5	GTCATCTTAGTCATTACCTGAGG	chr1:185056767-185056789	RNF2	0	[6]	0	0
6	GGAATCCCTTCTGCAGCACCTGG	chr11:22647332-22647354	FANCF	0	[6]	0	0
7	GTCACCTCCAATGACTAGGGTGG	chr2:73161066-73161088	EMX1 target 1	1	[14]	1	0
8	GACATCGATGTCTCCCATTTGG	chr2:73161046-73161068	EMX1 target 2	0*	[14]	0	1
9	GCGCCACCGGTTGATGTGATGGG	chr2:73161007-73161029	EMX1 target 6	0*	[14]	1	0
10	GTTGTGCTCAGTACTGACTTTGG	chr17:29483035-29483057	NF1_sg1	1	[15]	0	1
11	AGTCAGTACTGAGCACAACAAGG	chr17:29483039-29483061	NF1_sg3	0*	[15]	0	0
12	TTTCAGCTTCCAATAAAACAGG	chr17:29482995-29483017	NF1_sg4	0*	[15]	0	0
13	AAACATCTCGTACAGTGACAAGG	chr22:30057304-30057326	NF2_sg1	0	[15]	0	1
14	ATTCCACGGGAAGGAGATCTTGG	chr22:30057273-30057295	NF2_sg2	1	[15]	0	1
15	GTACTGCAGTCCAAAGAACCAGG	chr22:30032801-30032823	NF2_sg4	1	[15]	0	1
16	CACAGTGGCCTGGCTCAAAATGG	chr22:30032835-30032857	NF2_sg5	0*	[15]	0	0
17	AGGATTGAAGCTGACGTTCTTGG	chrX:70339299-70339321	MED12_sg1	0*	[15]	0	1
18	CGTCAGCTTCAATCCTGCCAAGG	chrX:70339306-70339328	MED12_sg2	1	[15]	0	1
19	CTCAGAGATTGCTGCATAGTAGG	chrX:70339961-70339983	MED12_sg3	0*	[15]	0	1
20	ACAGGTCATCTTAATGAGCCAGG	chrX:70339937-70339959	MED12_sg4	0*	[15]	0	1
21	GGGGCTGTGGTTCCACGATAGGG	chrX:70340905-70340927	MED12_sg5	0*	[15]	1	0
Totals:				9		3	9

**Table S4: Cas9-targeted sites characterized for off-targets in human genome** used to evaluate sensitivity and error profile of CasFinder's ability to identify targets that lack off-targets (i.e., 'specific' targets). A 1 in the "off-target" column indicates that SP Cas9 was found clearly to be active at at least one site in the human genome that is computationally similar to the target site. An asterisk (\*) indicates the possible presence of low SP Cas9 activity at a computationally similar site that appears to be at noise levels and is ignored here. Site 4 evaluated in ref. [6] was also evaluated in ref. [14] with similar results; only the ref. [6] evaluation is included in this Table. In the "CasFinder accept?" column, NA (Not Applicable) means that the target was not evaluated as part of the SP Cas9 hg19 human exome target set; otherwise, a 1 indicates that the target was evaluated and accepted, and a 0 indicates that it was rejected. The "In GECKO?" column indicates the presence of target in the published "GECKO" exome-wide target set generated in ref. [15], while a 0 means that it does not appear there. Targets that were evaluated but not accepted were not published in ref. [15], and insufficient information exists to clearly identify those which should be counted as NA.

a True Positive (TP), and any site characterized as *having* an off-target that CasFinder evaluated and accepted to be a False Positive (FP), where the CasFinder site evaluations were those performed when generating our human exome SP Cas9 target set. Similarly sites *with* off-targets that were evaluated and rejected by CasFinder are True Negatives (TN), and sites *without* off-targets that were evaluated and rejected by CasFinder are False Negatives (FN). The relevant numbers are then:

	P	N
T	2	6
F	1	10

Based on these values we can estimate CasFinder performance as follows:  $FDR = FP / (TP + FP) = 33\%$ ,  $sensitivity = TP / (TP + FN) = 17\%$ , and  $specificity = TN / (TN + FP) = 86\%$ . These values correspond to a relatively conservative algorithm, with high specificity and low sensitivity. As a partial comparison with another algorithm, we consider the target selection that was used to choose the GECKO set of exonic targets in ref. [15]. Here we cannot compute full statistics because only the “positive” targets actually chosen for the GECKO set were published, providing insufficient information to evaluate the “negative” set of targets that were considered but rejected. Thus only an estimated FDR can be calculated for GECKO: Of the 9 targets in the ‘accepted’ set, 4 had off-targets, corresponding to an FDR of  $4/9 = 44\%$ .

Caveats regarding these estimates include:

1. The set of 21 sites in [Table S4](#) is small and not representative of the many kinds of sequences that Cas9 will be used to target in human cells. Of the 21 sites, 19 are in exons from a total of 7 different genes, and the other two are from an upstream and an intronic region of one of the 7 genes. A better set of exonic sites would include sites from a wider set of genes and sets of genes with varied functions, but Cas9 site finding algorithms will also be applied to upstream and downstream regions, introns, non-coding RNAs, and enhancers and other sorts of regulatory regions that are present only as single instances or not at all in [Table S4](#). These regions will have different characteristics from exonic regions that will directly affect the profiles of similar sequences in the genome that algorithms use to evaluate the presence of off-targets, and these target-similar regions may also have different chromatin profiles that may affect the actual levels of activity of Cas9 in these locations.
2. As discussed in [Comparison of the CasFinder system with other Cas9 target evaluation algorithms](#) below, computation of a Cas9 target set is always a practical exercise in which Cas9 target specificity must usually be balanced against other priorities. This makes the estimation of FDR for the GECKO target set particularly problematic, since the object in ref. [15] was to pick an exome-wide target set that, to get sufficient coverage of human genes, likely required the use of liberal specificity scoring thresholds. Similarly, as emphasized in our article, CasFinder was designed with default scoring parameters and options that emphasize specificity, and is built to enable users to easily adjust these options to find sites in regions that are hard to target at these default levels.
3. It is also worth noting that Cas9 off-target activity profiles will likely be different in different human cell lines, may depend on the conditions under which Cas9 and Cas9 sgRNAs are expressed, and will also likely differ across different Cas9s.

Ultimately these limitations are due to the fact that it is difficult to detect off-targets of any genome editing method in a systematic way. Only a few methods exist capable of quantitatively detecting off-targets genome-wide in cells *in vivo* [16,17], the most appropriate data to use as a ‘ground truth’ basis for assessing Cas9 site evaluation algorithms. These methods are complex to conduct and have only been applied in a few instances, and so far as we know, none have been used on Cas9. Most information on Cas9 specificity has been gathered using reporter assays, testing sgRNAs with small numbers of mismatches to a particular target, or by *in vitro* methods [8,14,18]. These methods have been very useful for identifying the general rules regarding Cas9 specificity that have been incorporated into Cas9 site evaluation algorithms, but they do not provide data at the level of ‘ground truth’ for actual Cas9 off-targeting encountered during the targeting of endogenous sites in cells *in vivo*. The only data available at this level are from the types of studies described in [Table S4](#), where sites computationally

identified as similar to an explicitly targeted site are individually assayed for off-target activity, a low throughput process. While it can be assumed that means will eventually be developed to find off-targets to particular programmed targets in a high-throughput manner, it may yet take additional innovation to develop methods that can do this for many programmed targets in multiplex.

Beyond overcoming these limitations to throughput, several other steps can be identified that need to be addressed so that the assessment of sensitivity and accuracy of Cas9 site evaluation algorithms can be conducted fairly and robustly:

1. A database should be created and curated in which data on target sites that have been evaluated for off-targets can be assembled and curated in standardized, computer-readable form.
2. Representative sets of targets should be identified in regions of different types that can be used to evaluate algorithmic performance in such regions as exons, introns, upstream and downstream regions, in non-coding genes of different classes, and in genes in different functional categories. Experiments gathering data on actual off-targeting for these representative target sets should then be encouraged.
3. Standards for cell lines, and for Cas9 and sgRNA expression, should be suggested to promote uniformity in gathering data on off-targeting. However, the database in 1 should be designed to admit variation from such standards. The database should also support a variety of Cas9 proteins.
4. Efforts should be made to encourage Cas9 site evaluation algorithms to be made available in program form so that they can be run on the target sets of 2 as these sets are developed, and so that their program code can be examined and run by researchers evaluating these algorithms on their own computers. As noted in [Comparison of the CasFinder system with other Cas9 target evaluation algorithms](#), many algorithms have not been made available as programs, and others are only accessible through web sites so that their program code is not available for inspection.

It may be asked whether the sensitivity and error rate of CasFinder identifications of putatively 'specific' sites might be estimated by randomization, for instance, by evaluating sites accepted by CasFinder on a real genome again on a shuffled genome to see if they continue to be accepted. This would be an appropriate strategy if the CasFinder algorithm were a classifier or discriminator that had been optimized on a training set of 'ground truth' specific and non-specific sites, but CasFinder is primarily a sequence search algorithm that, for each of a set of candidate sites, searches the genome for other sequences that closely match the candidate and that are followed by a PAM motif. Running CasFinder on a shuffled genome would simply search the randomized genome for sequences matching the sites previously accepted in a real genome, and accept or reject them based on the presence of these essentially random matches. This would not be informative about the ability of the algorithm to distinguish specific from non-specific sites in the real genome.

<b>Using CasValue to re-screen CasFinder targets for higher specificity</b>
---

As noted in the main article, when CasValue searches for potential off-targets to Cas9 candidate target sequences, it defaults to searching the genome for matches to the target's seed sequence that contain up to 1 mismatch. The rationale for this is that the Cas9 activity is sensitive to mismatches in the seed region and that sites with 2 or more seed mismatches have low potential for off-target activity; hence retrieving and evaluating such sites can be considered dispensable [6,8,14,18]. However, off-target

activity at such sites is still possible [6,8]. Moreover, for performance reasons (see [CasFinder system performance considerations](#)) CasValue is also set up to assume that seeds are 13bp, longer than generally assumed, so that some of the 2 bp mismatching seed sites in the genome that CasValue ignores may support higher levels of Cas9 off-target activity. In applications where one needs greater assurance that such off-targeting will be minimized, it may be desirable to run CasValue with parameters that stipulate higher stringency specificity constraints. The main ways to stipulate higher specificity are to have CasValue assume shorter seed lengths than 13bp, or have it search for seed matches with 2 or more mismatches, as follows:

1. To cause CasValue to use shorter seed lengths, one can increase the parameter that defines the position of the seed start within a Cas9 targeting site. This parameter is set in the CASFINDER\_CAS9\_CONFIG, which, in the version distributed with this article, assumes that Cas9 targeting sequence lengths are 20bp and that seeds start in position 8 and end in position 20 (see [Figure S4](#)). To specify a seed length of 12bp therefore entails resetting the seed start position to 9. This may be done in either of two ways:
  - a. Use a copy of the CASFINDER\_CAS9\_CONFIG file in which the seed start position has been increased to 9 (see [Figure S4](#)). [Note that unless multiple versions of the CASFINDER\_CAS9\_CONFIG file are set up, this would change the seed length for *all* executions of CasFinder and CasValue. If multiple versions are set up, the one that any particular CasFinder or CasValue execution will use may be controlled by use of the `-CONFIG_CAS9` option (see [Table S2](#) and [Table S3](#)).
  - ... or ...
  - b. Run CasValue with the `-ss` option (see [Table S3](#)) to override the CASFINDER\_CAS9\_CONFIG specification of the seed start for a particular CasValue execution. [This method leaves the CASFINDER\_CAS9\_CONFIG unchanged.]
2. To cause CasValue to search for seed matches with 2 mismatches, one can run CasValue with the `-sm 2` option set to 2 (see [Table S3](#)).

In addition to having these two different options for stipulating higher specificity, one can also employ these options in two different ways. (i) One can employ them when one runs CasFinder to perform a search for Cas9 sites in specified input sequences, or (ii) one can run CasFinder first with default (lower stringency) options to get an initial set of Cas9 target sites in these input sequences, and then run CasValue subsequently with the higher specificity options to filter the initial set for more specific targets. *We strongly encourage method (ii) and recommend that it be applied selectively to only those initial sites of most interest.* The reason is the impact of higher stringency parameters on performance. For sake of illustration, assuming a 3Gb genome in which all bases have equal frequency, CasValue's default parameters of 13bp seeds and 1 mismatch seed searches should retrieve an average of 1788 genome matches per candidate Cas9 target. Changing parameters to search for 12bp seeds with 1 mismatch would retrieve an average of 6616 matches per query (3.7x the default), 13bp seeds with 2 mismatches would retrieve 33170 (18.5x), and 12bp seeds with 2 mismatches would retrieve 112832 (63.1x). Since most candidate Cas9 targets found by CasFinder are rejected with the default parameters (e.g., between 74.0% and 95.6% for the genomes and Cas9s in Table 2 of the main article), using method (i) by stipulating higher specificity in the initial CasFinder search would incur these performance penalties needlessly for a considerable majority of candidate targets. In contrast, using method (ii) only incurs them for the smaller number of sites that have passed the default specificity constraints. This can be further reduced if only selected subsets of sites of interest are evaluated with stringent parameters. It

might also be necessary evaluate smaller subsets because bowtie can require very large amounts of memory when processing shorter queries and multiple mismatches.

To illustrate procedure (ii), we will assume that we want to check the first site found in the search specified in our CasFinder system installation test (see section [Testing the installation](#)). This is a site in the first exon of TP53 at chr17:7590702-7590724 (hg19 coordinates) with the sequence AGGGAAGCGTGTCACCGTCGTGG. To evaluate this site with higher specificity parameters, the *targeting sequence* within this site (i.e., *excluding the PAM*) must be put into a CasValue input FASTA file in the form:

```
>TP53_exon_1_site_1 accept=1
AGGGAAGCGTGTCACCGTCG
```

(Note that accept=1 is the CasValue default so this may be omitted if 1 this is the desired ‘accept’ number for the site; see section [CasValue input file specifications](#).) CasValue may then be run as indicated in [Table S5](#). The site will be understood to have passed the higher stringency specificity checks *if it appears in the CasValue output file* (see section [CasValue output files](#).) In the CasValue commands given in [Table S5](#), the CasValue input file will be assumed to have the name *in.fa*, and the output file prefix will be assumed to have the name *out*.

option	Parameters		CasValue command	Retrieved matches				Pass?
	Seed Inth	Seed mms		Estimated		Actual		
				N	fold	N	fold	
1	default		perl CasValue_v2 -i in.fa -o out	1788	1.0	123	1.0	Y
2	12	1	perl CasValue_v2 -i in.fa -o out -ss 9	6616	3.7	1070	8.7	Y
3	13	2	perl CasValue_v2 -i in.fa -o out -sm 2	33170	18.5	4642	37.7	Y
4	12	2	perl CasValue_v2 -i in.fa -o out -ss 9 -sm 2	112832	63.1	38108	310	N
5	20	3	perl CasValue_v2 -i in.fa -o out -ss 1 -sm 3	89	.05	20	0.2	N

**Table S5: Use of CasValue to screen sites for higher specificity.** Options, corresponding CasValue commands, and results are shown that illustrate how CasValue can be used to determine if a Cas9 site found previously by CasFinder would pass increased specificity constraints. Option 1 represents the default constraints already imposed in the CasFinder run that initially found the site: seed length 13bp and retrieval of all seed matches in the genome with up to 1 mismatch from the target sequence’s seed. Options 2-4 illustrate the execution and outcome of CasValue re-evaluation of the site with seed length 12 and/or searching and evaluating seed matches with up to 2 mismatches from the target’s seed. Option 5 is discussed in the text below. “Retrieved matches” presents the number of seed matches retrieved from the genome, either “Estimated” as described in the text (i.e., assuming a 3e9 base genome and equal probability bases), or “Actual” (i.e., actually retrieved for the TP53\_exon\_site\_1 sequence described in the text for SP Cas9 and the hg19 human genome); here, “N” represents actual numerical values (or the rounded average, in the Estimated case), and “fold” is the fold change relative to the default (option 1). “Pass?” indicates whether the site passed the option’s specificity constraints: Y = yes, N = no. The **highlight** indicates that option 2 is recommended (see discussion in text).

As can be seen in [Table S5](#), the different options discussed above result in retrieval and examination of higher numbers of seed matches in the genome as specificity is increased, and the number of retrievals grows with increased specificity in the same order as estimated in the text above (although actual numbers deviate considerably from the initial probability-based estimates). The highest specificity option discussed above (option 4) – seed length of 12 and retrieval of matches with up to 2 mismatches

per seed – results in ~310 times the number of retrievals as obtained for the site in the CasFinder run that identified the site (represented by option 1). It can also be seen that at the level of specificity offered by option 4, the site is actually rejected, while it continues to be accepted at the intermediate levels of specificity of options 2 and 3. This illustrates the point discussed in the section on *Usage Considerations* of the main article that specificity checking must be balanced against the need to find sites in a region of interest.

The question of whether and how to re-screen sites found with CasFinder default options with higher specificity ultimately rests on how and how often the default options might cause CasValue to overlook similar genomic sites that could be important. With its default options, CasValue will ignore any sites with 2 mismatches in the seed region, and using the `-sm` parameter to retrieve of sites with *more* seed mismatches (options 3 and 4 in [Table S5](#)) will cause return of many of these ignored sites. However, this is not necessarily a good strategy as it is known experimentally that the Cas9 activity is particularly sensitive to seed region mismatches [6,8], so scrutiny of these new 2bp mismatching seed sites might cause undue rejection of a target. For instance, a genomic site whose full Cas9 footprint has two mismatches close to the PAM and otherwise matches a Cas9 candidate target site will be returned and evaluated at `-sm 2` and cause rejection of the candidate target, even though such a site has low off-target potential. This site would not be seen at the default `-sm` value of 1.

For these reasons, we recommend strategies such as option 2 in [Table S5](#) that examine sites using shorter seed sequences. From the perspective of ‘overlooked’ genomic sites that are brought under scrutiny, changing the seed to a shorter length causes CasValue to retrieve *some* new genomic matches, but not matches with multiple mismatches near the PAM as described above that have low off-target potential. The ‘overlooked’ sites that will be returned will be those with a mismatch distal to the PAM: For instance, a candidate Cas9 target site with a genomic match with 2 mismatches to its last 13bp (not including the PAM), where one of the mismatches is at the position furthest from the PAM, will not be retrieved with the default 13bp seed length, but will be retrieved if CasValue is run stipulating 12bp seeds. [Table S6](#) provides information on the number of new matches that can be retrieved under these circumstances and the percent of sites accepted at default settings that would be rejected with the shorter seed lengths. The results again indicate the trade-off that exists between stringency of specificity and the ability to find sites. Even though many sites may be rejected from the point of view of 12bp seeds that were accepted with the default settings, it should also be remembered that the ‘overlooked’ genomic sequences brought under scrutiny at these shorter seed lengths and cause initially accepted sites to be rejected, all still have at least 2bp mismatches with the sites’ targeting sequences, and most of these still contain single mismatches within the shorter seeds.

Finally, we note that if it is desired to retrieve and evaluate *all* potential off-target sites with  $\leq 3$  mismatches to a candidate target, one can run CasValue with the parameters specified by option 5 in [Table S5](#). However, although this leads to a very rapid and comprehensive evaluation, it is also unreasonably conservative as it would even cause rejection of a candidate target if there were a single sequence in the genome with 3 mismatches adjacent to the PAM, which should essentially abrogate Cas9 activity.

Non-default seed length	Percent of sites accepted at seed length 13 rejected at specified seed length	Number of Cas9 footprints meeting score threshold retrieved at specified seed length not retrieved at seed length 13
12	26.0 $\pm$ 0.4	0.45 $\pm$ 6.41
11	44.5 $\pm$ 0.3	0.98 $\pm$ 15.73
10	57.0 $\pm$ 0.4	1.54 $\pm$ 28.39

**Table S6: Numbers of sites accepted at default CasValue settings that would be rejected at smaller seed lengths.** Three sets of 5000 Cas9 targets were picked at random from our full SP Cas9 human exome site catalog and analyzed for how many would be rejected had potential off-targets been retrieved using seed lengths smaller than the default seed length of 13. Queries for genome matches at each seed length returned all matches in the hg19 human genome with up to 1 mismatch. Column 2: Percentages of rejected sites at the specified seed length were computed for each of the three sets of 5000 sites; values represent the mean  $\pm$  1 stdev of these three percentages. Column 3: Values represent the mean  $\pm$  1 stdev over all 15000 sites of Cas9 footprints meeting the  $-x$  threshold of 3 at the specified seed length that were not retrieved and scored at default seed length 13.

#### CasFinder system performance considerations

To ensure successful operation, the CasFinder system must be provided with appropriate computer resources and run on appropriately-sized input sequence data sets. CasFinder and CasValue program options can also be used to modulate performance. This section reviews the main considerations governing resource utilization and performance, drawing on empirical data on performance provided in Table 2 of the main article, and offers advice on how to control performance.

#### Memory requirements

Memory requirements are generally the most important resource requirements for CasFinder operation and are mainly determined by the following factors:

1. CasFinder maintains in memory: (i) all keyword, annotation, and sequence specified in its input file (see section [CasFinder input file specifications](#)), (ii) tables tracking each candidate Cas9 site found in the input sequence. Note that these tables are maintained in memory during CasFinder's call to CasValue.
2. CasValue maintains in memory: all keyword, annotation, and sequence specified in its input file (see section [CasValue input file specifications](#)), and so duplicates some of the memory requirements CasFinder incurs for candidate Cas9 targets. For each Cas9 targeting sequence it evaluates, it maintains a table of lowest scores of potential off-targets in the genome, whose size is governed by the targeting sequence's 'accept' number. Memory of keyword, annotation, and input sequences is maintained during CasValue's call to bowtie.
3. Bowtie's memory requirement depends on the size of the  $-g$  genome, as bowtie must load its FM-index into memory [5].
4. CasValue and CasFinder both load  $-g$  genome chromosome sequences into memory in the course of their operations. CasValue loads chromosome sequences in the course of extracting Cas9 footprints from the genome based on seed matches returned by bowtie. CasFinder loads

the sequences when processing \*EXTRACT\* statements in the CasFinder input file. In both cases, in an effort to avoid loading entire genomes into memory, the programs batch all extract requests by chromosome and process each chromosome once and one at a time. However, this means that both programs must accumulate tables of all extract requests in memory before loading the chromosome sequences, and these can be very large for CasValue if the number of matches returned by bowtie is large.

While all of these requirements count, CasValue's internal table of bowtie match locations in item 4 is usually the main determinant of the maximum memory needed for successful operation. By and large this is in turn controlled by:

- The amount of sequence specified in the CasFinder input file, as this controls the number of candidate Cas9 target sites evaluated by CasValue.
- The size of the -g genome, which affects the number of matches returned by bowtie to seed sequence queries.
- CasFinder and CasValue program options determining how bowtie does its search, including the use of the bowtie -k option (which limits the number of bowtie matches returned per query), and the CasFinder -sm option (which determines the number of mismatches within bowtie matches). The size of the seed region in the Cas9 targeting sequences, as specified by the CASFINDER\_CONFIG\_CAS9 file can also affect the number of bowtie matches returned per query. (See the sections on [CasFinder](#) and [CasValue program options](#), as well as the section [3.b Updating the CASFINDER CAS9 CONFIG file](#), for further information.)
- Processing of candidate Cas9 targets in repetitive sequence can have a large impact on the number of bowtie matches returned per query, and can also affect the number of candidate Cas9 targets found in the CasFinder input sequences. For this reason, the CasFinder is set up by default to not process candidate Cas9 target sequences containing repetitive sequence (see the sections on [CasFinder input file specifications](#) and [CasFinder program options](#) for details). However, even when repeatmasked sequence is avoided, some seed sequences can generate extremely large numbers of bowtie matches. In one test of bowtie retrievals from 10000 randomly selected, non-repeatmasked human exonic 13mers (the size of seed sequences specified for this study) with up to one mismatch, one 13mer generated 692180 returned matches (mean = 6542.9, stdev = 16770.4).
- The *max memory* data in Table 2 of the main article on CasFinder runs in the human and mouse exomes illustrates an additional dependency of memory on the particular Cas9 being processed. It can be seen that in terms of memory requirements, NM > SP > ST1, and this is also the order seen in *bowtie matches*. This order reflects the relative degeneracy of the PAMs of these three Cas9s (see Table 1 in the main article text), with NM being the most degenerate and ST1 being the least. The probably reason for this memory requirement relationship is that CasFinder will on average find more candidate Cas9 sites in the same input sequences for Cas9s with more degenerate PAMs, than Cas9s with less degenerate PAMs (see *candidate targets* in Table 2 of the main article), and will therefore generate more seeds and bowtie queries for these Cas9s.

Table 2 of the main article shows how these factors were accommodated and balanced during the generation of Cas9 sites for the human and mouse exomes (see below):

- Most importantly, the exomes were divided into parts each of which presented less than a specified ceiling of sequence size to CasFinder. We found that CasFinder input files presenting  $\leq 500\text{kbp}$  in sequence could be manageably run within the resource limits described in Table 2 of the article.
- The bowtie  $-k$  parameter was used to limit the number of bowtie matches returned per query. As can be seen in Table 2 of the main article,  $-k$  values of 20000-25000 were selected. Values were chosen so that the fraction of accepted Cas9 targets for which the *all\_matches\_analyzed* indicator was 0 was  $< 1\%$  over the entire exome. For SP and NM Cas9s,  $-k$  values of 20000 were sufficient, but for ST1, a  $-k$  value of 25000 was required. This may be due to the relatively low degeneracy of the ST1 PAMs, which makes potential off-targets to candidate ST1 Cas9 sites rarer, and which may in turn require CasValue to inspect more bowtie returns per query to find off-targets that cause the candidate sites to be rejected.

Under these conditions, as seen in Table 2, CasFinder required  $< 31000\text{MB}$  memory to find SP Cas9 sites,  $< 12000\text{MB}$  memory to find ST1 sites, and  $< 56000\text{MB}$  memory to find NM sites, in  $\sim 500\text{kbp}$  blocks of human and mouse repeatmasked exonic sequence.

#### *CPU time requirements*

These are generally secondary and parallel to memory requirements. Generally speaking, the processing segment that requires the longest time in CasFinder and CasValue operation is CasValue's extraction and analysis of Cas9 footprints from the chromosome files based on bowtie matches found to seed sequences (item 4 in *Memory requirements* above). It can be seen from Table 2 of the main article, that to process  $\sim 500\text{kbp}$  of human or mouse repeatmasked exonic sequence required on average  $< 03:20$ ,  $00:57$ , and  $05:35$  (hours:minutes) for SP, ST1, and NM Cas9s, respectively.

*NOTE: Related to the fact that extraction and analysis of Cas9 footprints around bowtie matches from chromosome files takes the largest amount of processing time of all steps in CasFinder system operations, is the observation that this step can also cause long processing times even when the system is working with very small amounts of input sequence. This is because even a single candidate Cas9 site in input sequence can generate bowtie matches all across the  $-g$  genome and so require time-consuming loads of many or all chromosome files.*

#### *Temporary file storage requirements*

As noted in sections [3.a Updating the CASFINDER CONFIG file](#) and [CasValue operation](#), the CasFinder system is set up to use temporary files when passing input to CasValue, and CasValue uses temporary files in its call to bowtie. By far the main file temporary storage requirement is bowtie's generation of its output file, as the number of bowtie returns to  $\sim 500\text{kbp}$  can be extremely large, even when the bowtie  $-k$  option is in use: Table 2 of the main article shows that the number of bowtie matches returned on average to  $\sim 500\text{kbp}$  of CasFinder input human or mouse repeatmasked exome sequence is approximately  $3.1\text{e}8$ ,  $1.0\text{e}8$ ,  $5.4\text{e}8$  for SP, ST1, and NM Cas9s, respectively. The temporary files accommodating this output can be  $> 30\text{GB}$ . These observations highlight the importance of the note in section [3.a Updating the CASFINDER CONFIG file](#) on specifying a *temp\_file\_prefix* that is in a */tmp* directory that is regularly cleared out, so that these files do not accumulate.

<b>Generation of the all human and mouse exome catalogs of Cas9 sites</b>
---

### *Definition of the human and mouse exomes*

CasFinder input files were generated for the human and mouse exomes (versions GRCh37/hg19 and GRCm38/mm10, respectively) based on information in the UCSC Table Browser knownGene and kgXref tables [1,2,13]. Exon annotations were grouped and annotated by geneSymbol using the following procedures.

1. The complete contents of knownGene, joined with geneSymbol and refseq information from kgXref, were downloaded from the UCSC Table Browser on October 23, 2013. In these annotations, many transcript isoforms (designated by UCSC accessions) typically map to a geneSymbol, and it may also happen that accessions assigned to a geneSymbol may be found mapped to more than one location in the genome (i.e., multi-copy genes).
2. First, bedtools [19] was used to identify multi-copy genes: For each geneSymbol, a BED file was generated from all UCSC accessions associated with the geneSymbol using the chrom, txStart, and txEnd to define the accessions' locations. The bedtools merge function was applied strand-specifically, resulting in distinct groupings of accessions in disjoint genome locations. geneSymbols which were found to have multiple copies were then given modified names to distinguish these copies, where the modification was to add the notation (copy:n) to the geneSymbol ( $n=1,2,\dots$  up to the number of identified copies). For instance, the NAIP gene was found to have two copies in close proximity on chr5, which were distinguished as NAIP(copy:1), and NAIP(copy:2), respectively. The names of geneSymbols not found to have multiple copies in the genome were left unchanged.
3. Then, bedtools was used again to merge the exons for all UCSC accessions mapped to the (now copy-distinguished) geneSymbols to yield distinct *exon\_groups*. As in 2, for each copy-distinguished geneSymbol, a BED file was generated for all exon locations within the mapped accessions using chrom, exonStart, and exonEnd as the exon locations, and the bedtools merge function was again used to strand-specifically merge these regions. As generated, these merged regions often represented the overlaps of the same exon as it appeared in different annotated isoforms, where that exon could have different numbers in the different isoforms. For instance, the same exon might be exon 3 in one isoform but 4 in a second, where in the first isoform a prior exon was skipped. Thus, the merged regions could not be assigned any determinate exon number, and were instead identified as *exon\_groups* assigned to the copy-distinguished geneSymbols, which were designated by suffixing the copy-distinguished geneSymbol with (exon\_group:1), (exon\_group:2), etc., up to the number of *exon\_groups*, and where the numbering 1,2,... was assigned in 5' to 3' order based on the strand of the copy-distinguished geneSymbol. For instance, the NAIP(copy:2) geneSymbol was found to have 19 *exon\_groups*, and within these, *exon\_group* 6 (designated NAIP(copy:2):(exon\_group:6)) was a merge between exon 4 of UCSC gene uc003kar.1 and exon 2 of UCSC gene uc011crs.1.
4. Finally, CasFinder input files were built by generating CasFinder \*EXTRACT\* input records for each *exon\_group* computed in 3, padding the merged exonStart and exonEnd locations by 20bp on each side. The 20bp padding was included to enable CasFinder to find Cas9 sites targeting the very beginning and end of each exon. (Note, however, that given that the dsDNA cuts generated by Cas9 are in the 3' end of the targeting sequences [for SP Cas9, at any rate] [4], Cas9 sites that target the padding regions may only cut the within exon sequence when oriented *towards* the gene.)

Files describing the sequence ranges of each *exon\_group* (with padding) of each (copy-distinguished) geneSymbol, and what UCSC and RefSeq annotations were merged into them, have been included with the distribution of our catalogs of exome Cas9 sites on <http://arep.med.harvard.edu/CasFinder>. The two files are identified as *gene\_exon\_index* files and contain the following columns:

1. *gene*: The geneSymbol, with a (copy:n) designation if it the gene is multi-copy
2. *gene\_location*: The sequence range associated with the gene copy that resulted from the bedtools merge of txStarts to txEnds of geneSymbol-associated UCSC gene records used to disambiguate the locations of multi-copy genes.
3. *gene\_strand*: The orientation of the gene copy.
4. *gene\_type*: This field should be ignored. (It represented an attempt to identify whether the gene was coding or non-coding by counting the number of UCSC gene records merged to the geneSymbol that had either had, or had no, associated protein accessions, but this was not successful.)
5. *exon\_group*: The *exon\_group* number we assigned to a merged exon region.
6. *exon\_group\_range*: The sequence range of the merged set of exon regions that comprised this *exon\_group*, including 20bp of padding at each end.
7. *associated\_refseq\_ids*: A list of any RefSeq IDs that were annotated as associated with the UCSC gene records that were merged into the *exon\_group*. Different RefSeq IDs could be found associated with different UCSC gene records sharing an exonic region, and, likewise, the same RefSeq ID could be found associated with multiple UCSC gene records.
8. *number\_UCSC\_exons\_in\_group*: The number of exon ranges from UCSC gene records that were merged together into the *exon\_group*.
- 9+ *UCSC\_exons\_in\_group...*: *number\_UCSC\_exons\_in\_group* columns now follow, each identifying the exon of a UCSC gene that was merged into this *exon\_group*. For the hg19 exome, this ranged between 1 and 134, and for the mm10 exome, this ranged between 1 and 49.

#### *Generation of the Cas9 site catalogs for each exome*

After assessing CasFinder performance as described above (see [CasFinder system performance considerations](#)), we divided the exome-wide CasFinder input files into parts by sequentially accumulating \*EXTRACT\* records in groups that in aggregate specified  $\leq 500$ kbp, starting a new group whenever this limit was exceeded. As noted in Table 2 of the article, this resulted in 197 parts for the hg19 genome, and 177 for the mm10 genome. CasFinder was then run on each of these parts for each of the three Cas9s in parallel to the degree possible on Harvard Medical School's Orchestra High Performance Compute Cluster. The 197 or 177 CasFinder output *data* and *stat* files were then aggregated for each genome and Cas9 using perl scripts (see [CasFinder output files](#)).

#### *Generation of reduced set of Cas9 targets for gene knockout screens*

To facilitate the construction of genome-wide libraries of Cas9 sgRNAs targeted to knock out genes, we selected up to three Cas9 targets per gene from the full exome Cas9 site catalogs. Targets were chosen using a heuristic that attempted to strike a balance between picking targets in the 5'-most regions of each gene (which have potential to generate frameshifts abrogating more of the protein than 3'

disruptions), against the possibility that, due to alternate transcription starts and splice variants, the 5'-most exons might not be expressed in all cell types or circumstances. Specifically:

1. For each *exon\_group* within each gene, we first sorted all Cas9 targets by the proximity of the dsDNA cut site to the 5' end of the exon, eliminating from consideration any targets overlapping the 20bp 5'-padding added to the *exon\_group* sequence whose orientation placed the cut outside of the exon itself (see point 4 in the section [Generation of the all human and mouse exome catalogs of Cas9 sites](#) above). Note that while SP Cas9 cut sites are known to be blunt-ended and between the 3<sup>rd</sup> and 4<sup>th</sup> bp upstream of the PAM [4], ST1 and NM cut sites have not been characterized. We assumed that ST1 and NM Cas9s generated the same kind of cut sites as SP.
2. For each gene, we then took the ordered Cas9 site sets from step 1 for the *three 5'-most exon\_groups* for which such sites existed, with the intent of picking the 5'-most cutting Cas9 site in each of these *exon\_groups*. However, many genes only had sites in one or two *exon\_groups*, not in three, and the number of sites in each *exon\_group* might be single or multiple. To accommodate these possibilities in a general fashion, we cycled through the (up to) three 5'-most *exon\_groups* in 5'-to-3' order, and picked out (without replacement) the 5'-most Cas9 site from each, until we had either picked three Cas9 sites (if there were three or more), or picked them all (if there were not).

For each Cas9, the procedure above generates sites in the reduced Cas9 target set for all genes for which there are any Cas9 targets at all, except where all of the targets in the full set have Cas9 cut sites that fall outside of all exons of the gene. The fractions of gene copies in each of our target sets that fail to have sites in their reduced sets for this reason is given in [Table S7](#). In all cases this fraction is  $\leq 1.8\%$ . The gene copies missing from the reduced sets can be identified in the files described in the section [Statistics at the gene copy level](#) below.

In applying the method above, we picked sites for each gene copy of multi-copy genes separately. Information on each reduced set of Cas9 targets was assembled into a modified form of the CasFinder *data* and *stat* output files (see [CasFinder output files](#)) that is indicated by the tag "select3" in the file names.

- Select3 *stat* files represent all the distinct gene copies listed in the original whole exome CasFinder *stat* files, but give information at the gene copy instead of the *exon\_group* level. The columns of the file are: (1) The *gene\_number* of the gene copy. These are assigned sequentially to the gene copies found in the CasFinder *stat* files; (2) the *gene* name; (3) *selected\_sites* indicates the total number of sites that were selected for the gene (which, by construction, must be  $\leq 3$ ); (4) *num\_exon\_groups* then gives the number of *exon\_groups* from which *selected\_sites* were picked. A pair of columns then follows *selected\_sites* for each of the *num\_exon\_groups* *exon\_groups* from which the sites were picked, the first of which identifies the number of the *exon\_group* itself, and the second of which gives the number of sites picked from that *exon\_group*.
- In select3 *data* files, the CasFinder *data* file records from the original whole exome Cas9 site catalogs for all picked sites are retained exactly but prepended with three new columns: (1) *target\_number* simply assigns numbers sequentially for each of the select3 sites; (2) *gene\_number* is the sequential *gene\_number* assigned to the gene copy for the current site in the select3 *stat* file above; (3) *gene\_id* is the gene copy name. As just noted, an exact copy of

the exome Cas9 catalog *data* file row for the selected site now follows, but columns (4) and (5) are renamed *xref\_target\_number* and *xref\_sequence\_id* to avoid confusion with the new column (1) and (2) *target\_number* and *gene\_id*. The *xref\_target\_number* and *xref\_sequence\_id* can be used to cross-reference the *data* file row to the original CasFinder output *data* files.

genome	hg19			mm10		
Cas9	SP	ST1	NM	SP	ST1	NM
gene copies	31613			32480		
gene copies with targets	24754	24974	24218	25176	25875	24217
gene copies with targets not in reduced set (%)	263 (1.1)	287 (1.1)	161 (0.7)	410 (1.6)	475 (1.8)	214 (0.9)

**Table S7: Numbers of gene copies with targets in the full Cas9 catalogs that do not appear in the corresponding reduced target sets.** This circumstance arises when all the targets for a Cas9 in the full catalog have cut sites in the padding regions outside of the exons. See text for details.

#### *Statistics at the gene copy level*

We have provided statistics on the number of sites at the gene copy vs. exon group level in two files: hg19\_exome\_12122013.gene\_sites.stat.txt and mm19\_exome\_12122013.gene\_sites.stat.txt. Each of these files gives the total number of Cas9 sites found over all exon groups of each gene copy in the full Cas9 target catalogs and in the reduced sets. Gene copies that have sites in the full catalog for a Cas9 but are missing from the corresponding reduced set can be identified as those whose values are > 0 in the Cas9 column but are equal to 0 in the corresponding “select3” column.

### **Comparison of the CasFinder system with other Cas9 target evaluation algorithms**

While several algorithms for evaluating Cas9 targets have been described, they are heterogeneous as a group in purpose and design. Cas9 evaluation algorithms have most commonly been developed in connection with generation of genome-wide or exome-wide Cas9 target sets [3,15,20-23]. Although these find Cas9 targets in a genome and evaluate their specificity, they invariably integrate other criteria important to the screen, such as targeting to particular exons of a gene [15,22] (also, see [Generation of reduced set of Cas9 targets for gene knockout screens](#) above), filtering for particular restriction sites that will be used in library generation [22]. They may also relax specificity to permit targeting of genes with multiple copies or pseudogenes [22], or, alternatively, increase specificity constraints, particularly when processing smaller genomes where the computational burdens are reduced (examples are the use of reduced seed sizes in [20,23]). At the other end of the spectrum from exome- or genome-wide library generation, tools exist with the aim of finding the *native* targets of natural CRISPR systems (such as CRISPRTarget, ref. [24]) vs. finding targets in non-native genomes suitable for genome engineering. Thus the number of general purpose Cas9 target evaluation algorithms similar to the CasFinder system is small. To our knowledge, the methods closest to the CasFinder system are the CRISPR Design Tool [14] and the recently published CasOT [25] and ECRISP [26] systems, although even here CasOT is better described as a method for finding and reporting Cas9 site off-targets in a genome than a method for choosing targets based on the presence of possible off-targets.

A summary comparison of the CasFinder system with the CRISPR Design Tool and CasOT and with two recent efforts to generate human exome-wide target sets [15,22] is presented in [Table S8](#). A point of particular interest is that both the CRISPR Design Tool [14] and the “GECKO” library [15] were developed

by the same laboratory: Through both employ similar site scoring methods, they differ in many other ways and thus illustrate how evaluation methods may need to be modified when the focus changes from general purpose target finding to library design. Note that the “Predicted exome-wide sgRNA knockout library” described in [Table S8](#) is the second of two sgRNA libraries generated in the study [22]; the first is not described here. The heterogeneity of algorithm design for the methods covered in [Table S8](#) makes summary description difficult, but an attempt has been made to frame discussion in terms of the following common conceptual framework: Each method is assumed to

1. be directed to particular set of sequences in which candidate Cas9 targets will be analyzed (*Sequences in which CTs are evaluated*; note that the phrase “candidate Cas9 target” is abbreviated as CT in [Table S8](#));
2. allow the user to specify processing options (*Other input options*; this has been described as *Not applicable* to the library generation efforts in [Table S8](#));
3. find candidate Cas9 targets by searching for PAM motifs in the sequences of 1 above. However, at this time certain targets may be filtered out (*CT prefiltering*). (Note, however, that CasOT only performs a Cas9 target search in one of its three processing modes.)
4. search for possible off-targets to each candidate Cas9 target in a genome sequence of interest. However, this search may be restricted to only a subset of the genome (*CT OT search space*; note that the phrase “off-target” is abbreviated as OT in [Table S8](#));
5. having found possible off-targets for each candidate Cas9 target, the target’s specificity is evaluated by computations and comparisons particular to each method. The method may then select a subset of candidate targets based on the specificity evaluation, and some methods may then also filter out selected targets based on non-specificity criteria. (*CT evaluation*)
6. Finally, the method generates output files that identify the selected candidate Cas9 targets and summarize data generated during their evaluation (*Evaluation output*).

The comparison in [Table S8](#) supports the view that there are a wide variety of approaches and priorities in evaluating Cas9 targets, but also that little has been done on a software level to enable users to flexibly tune and tweak the selection of Cas9 targets to the needs of their research. The CasFinder system is offered as an initial entry into this niche that is both immediately useful and, we hope, can serve as a model for improving the architecture of and interface with Cas9 target analysis algorithms.

Evaluation method	CRISPR Design tool	“GECKO” sgRNA library	Predicted exome-wide sgRNA knockout library	CasOT	CasFinder system
Description	Web-based site selection tool	Design criteria for exome-wide sgRNA library	Design criteria for exome-wide sgRNA library	Software for finding OTs to specified CTs, or finding CTs in input sequence and their OTs	Software for finding Cas9 sites meeting specificity constraints
Reference	[14]	[15]	[22]	[25]	<i>this study</i>
Software available?	Available on web site only	No	No	Yes	Yes
Evaluation output	CTs in input sequence ranked by OT likelihood plus predicted OTs for each site	Exomic target library based on computed site specificity score	Predicted exomic target library based on site features	Three modes: (i) OTs to specified CTs; (ii) OTs to specified <i>paired nickase</i> CTs; (iii) CTs in input	List of CTs in user-specified sequence passing specificity criteria; optional scoring and OT site details

				sequence and their OTs	file
Sequences in which CTs are evaluated	User-specified	Constitutive human exons	Constitutive human exons	CTs specified by user for modes (i) & (ii); user-specified input sequence (< 1kb) for mode (iii)	User-specified
Other input options	Select 1 of 15 genomes	<i>Not applicable</i>	<i>Not applicable</i>	Genome sequence; PAM (NGG or NRG or NRG+NNGG, or no PAM); number of seed and non-seed MMs; target length	Cas9 and genome to process; Cas9 target site and PAM definitions; scoring weights; bowtie search parameters; CT acceptance criteria
CT pre-filtering	<i>ND</i>	CTs with OT $\leq$ 1MM from CT	GC-range & homopolymer filters; exclude CTs if match > 5 OTs (bowtie?)	None	Repetitive sequence (optional), T-strings (adjustable length)
CT OT search space	<i>ND</i>	All OT $\leq$ 3MM in genome. (Based on bowtie database of all PAM-suffixed in genome.)	<i>ND</i>	All strings ending in PAMs in user-provided genome sequence	All sites in genome with $\leq$ 1 MM (adjustable) with CT seed sequences; optionally limited by bowtie -k
CT evaluation	<i>ND</i> , but referred to ref. [14] Fig S11, where predicted target activities are computed as products of experimentally measured activities for sgRNAs with MMs at each position, adjusted with factor for MM clustering.	Each OT for a CT is scored by sum of MM positions, adjusted with factor MM clustering. The CT itself is scored as sum of its OT scores. For each exon, CTs with lowest scores < threshold are picked.	CTs are ranked by numbers of OTs in genome, numbers of OTs with 1 MM, the score of an SVM that predicts Cas9 activity, and factors relating to transcript location and number of transcript annotations. Overlapping CTs are filtered out and 5-10 CTs picked per gene.	For modes (i) & (ii), finds all OTs in to input CTs genome with $\leq$ specified MMs of specified PAM types. For (iii) first find CTs in input sequence with specified target length	For each OT a score is computed as a sum of seed and non-seed MM counts, each with user-specified weights, plus a secondary PAM cost. The CT reported to output if number of OTs with scores < -x threshold is $\leq$ input sequence 'accept' number.

**Table S8: Comparison of CasFinder system with other Cas9 target evaluation algorithms.** See text for discussion. Note that the "Predicted exome-wide sgRNA knockout library" described here is the second of two sgRNA libraries generated in the study [22], and that the first is not described here. *Abbreviations:* CT = candidate Cas9 target site; MM = mismatch between a candidate Cas9 target (CT) and one of its possible off-targets (OT); *ND*, not described in reference; OT = genome region considered a possible off-target to a candidate Cas9 target; SVM = Support Vector Machine.

## References

1. Dreszer TR, Karolchik D, Zweig AS, Hinrichs AS, Raney BJ, et al. (2012) The UCSC Genome Browser database: extensions and updates 2011. *Nucleic Acids Res* 40: D918-923.
2. Karolchik D, Hinrichs AS, Furey TS, Roskin KM, Sugnet CW, et al. (2004) The UCSC Table Browser data retrieval tool. *Nucleic Acids Res* 32: D493-496.
3. Mali P, Yang L, Esvelt KM, Aach J, Guell M, et al. (2013) RNA-guided human genome engineering via Cas9. *Science* 339: 823-826.
4. Jinek M, Chylinski K, Fonfara I, Hauer M, Doudna JA, et al. (2012) A programmable dual-RNA-guided DNA endonuclease in adaptive bacterial immunity. *Science* 337: 816-821.
5. Langmead B, Trapnell C, Pop M, Salzberg SL (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol* 10: R25.
6. Fu Y, Foden JA, Khayter C, Maeder ML, Reyon D, et al. (2013) High-frequency off-target mutagenesis induced by CRISPR-Cas nucleases in human cells. *Nat Biotechnol* 31: 822-826.
7. Jiang W, Bikard D, Cox D, Zhang F, Marraffini LA (2013) RNA-guided editing of bacterial genomes using CRISPR-Cas systems. *Nat Biotechnol* 31: 233-239.
8. Mali P, Aach J, Stranges PB, Esvelt KM, Moosburner M, et al. (2013) CAS9 transcriptional activators for target specificity screening and paired nickases for cooperative genome engineering. *Nat Biotechnol* 31: 833-838.
9. Ran FA, Hsu PD, Lin CY, Gootenberg JS, Konermann S, et al. (2013) Double nicking by RNA-guided CRISPR Cas9 for enhanced genome editing specificity. *Cell* 154: 1380-1389.
10. Esvelt KM, Mali P, Braff JL, Moosburner M, Yaung SJ, et al. (2013) Orthogonal Cas9 proteins for RNA-guided gene regulation and editing. *Nat Methods* 10: 1116-1121.
11. Maeder ML, Linder SJ, Cascio VM, Fu Y, Ho QH, et al. (2013) CRISPR RNA-guided activation of endogenous human genes. *Nat Methods* 10: 977-979.
12. Koike-Yusa H, Li Y, Tan E-P, Velasco-Herrera M, Yusa K (2013) Genome-wide recessive genetic screening in mammalian cells with a lentiviral CRISPR-guide RNA library. *Nat Biotechnol* advance online publication.
13. Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, et al. (2002) The human genome browser at UCSC. *Genome Res* 12: 996-1006.
14. Hsu PD, Scott DA, Weinstein JA, Ran FA, Konermann S, et al. (2013) DNA targeting specificity of RNA-guided Cas9 nucleases. *Nat Biotechnol* 31: 827-832.
15. Shalem O, Sanjana NE, Hartenian E, Shi X, Scott DA, et al. (2013) Genome-Scale CRISPR-Cas9 Knockout Screening in Human Cells. *Science*.
16. Chiarle R, Zhang Y, Frock RL, Lewis SM, Molinie B, et al. (2011) Genome-wide translocation sequencing reveals mechanisms of chromosome breaks and rearrangements in B cells. *Cell* 147: 107-119.
17. Gabriel R, Lombardo A, Arens A, Miller JC, Genovese P, et al. (2011) An unbiased genome-wide analysis of zinc-finger nuclease specificity. *Nat Biotechnol* 29: 816-823.
18. Pattanayak V, Lin S, Guilinger JP, Ma E, Doudna JA, et al. (2013) High-throughput profiling of off-target DNA cleavage reveals RNA-programmed Cas9 nuclease specificity. *Nat Biotechnol* 31: 839-843.
19. Quinlan AR, Hall IM (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 26: 841-842.
20. Li JF, Norville JE, Aach J, McCormack M, Zhang D, et al. (2013) Multiplex and homologous recombination-mediated genome editing in *Arabidopsis* and *Nicotiana benthamiana* using guide RNA and Cas9. *Nat Biotechnol* 31: 688-691.

21. Ren X, Sun J, Housden BE, Hu Y, Roesel C, et al. (2013) Optimized gene editing technology for *Drosophila melanogaster* using germ line-specific Cas9. *Proc Natl Acad Sci U S A* 110: 19012-19017.
22. Wang T, Wei JJ, Sabatini DM, Lander ES (2013) Genetic Screens in Human Cells Using the CRISPR/Cas9 System. *Science*.
23. DiCarlo JE, Norville JE, Mali P, Rios X, Aach J, et al. (2013) Genome engineering in *Saccharomyces cerevisiae* using CRISPR-Cas systems. *Nucleic Acids Res* 41: 4336-4343.
24. Biswas A, Gagnon JN, Brouns SJ, Fineran PC, Brown CM (2013) CRISPRTarget: bioinformatic prediction and analysis of crRNA targets. *RNA Biol* 10: 817-827.
25. Xiao A, Cheng Z, Kong L, Zhu Z, Lin S, et al. (2014) CasOT: a genome-wide Cas9/gRNA off-target searching tool. *Bioinformatics*.
26. Heigwer F, Kerr G, Boutros M (2014) E-CRISP: fast CRISPR target site identification. *Nat Methods* 11: 122-123.